Tracking Dynamic Regions of Texture and Shape

by

Joshua Migdal

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2007

Author .....
Department of Electrical Engineering and Computer Science
August 31, 2007

Certified by .....
W. Eric L. Grimson
Bernard Gordon Professor of Medical Engineering
Thesis Supervisor

Certified by .....
John W. Fisher III
Principal Research Scientist
Thesis Supervisor

Accepted by .....
Terry Orlando
Chairman, Department Committee on Graduate Students

Tracking Dynamic Regions of Texture and Shape

by

Joshua Migdal

Submitted to the Department of Electrical Engineering and Computer Science
on August 31, 2007, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

The tracking of visual phenomena is a problem of fundamental importance in computer vision. Tracks are used in many contexts, including object recognition, classification, camera calibration, and scene understanding. However, the use of such data is limited by the types of objects we are able to track and the environments in which we can track them. Objects whose shape or appearance can change in complex ways are difficult to track as it is difficult to represent or predict the appearance of such objects. Furthermore, other elements of the scene may interact with the tracked object, changing its appearance, or hiding part or all of it from view.

In this thesis, we address the problem of tracking deformable, dynamically textured regions under challenging conditions involving visual clutter, distractions, and multiple and prolonged occlusion. We introduce a model of appearance capable of compactly representing regions undergoing nonuniform, nonrepeating changes to both its textured appearance and shape. We describe methods of maintaining such a model and show how it enables efficient and effective occlusion reasoning.

By treating the visual appearance as a dynamically changing textured region, we show how such a model enables the tracking of groups of people. By tracking groups of people instead of each individual independently, we are able to track in environments where it would otherwise be difficult, or impossible.

We demonstrate the utility of the model by tracking many regions under diverse conditions, including indoor and outdoor scenes, near-field and far-field camera positions, through occlusion and through complex interactions with other visual elements, and by tracking such varied phenomena as meteorological data, seismic imagery, and groups of people.

Thesis Supervisor: W. Eric L. Grimson
Title: Bernard Gordon Professor of Medical Engineering

Thesis Supervisor: John W. Fisher III
Title: Principal Research Scientist

Contents

1 Introduction **11**

1.1 Why Track? 13

1.2 Tracking Paradigms 15

1.3 Overview of Thesis Work 17

1.3.1 Algorithm Parameters 19

1.3.2 Applications 20

1.3.3 Evaluation 20

1.4 Contributions 21

1.4.1 Joint model of shape and appearance 21

1.4.2 Group Tracking 23

1.4.3 Occlusion Reasoning 23

1.5 Outline of Dissertation 24

2 Tracking **25**

2.1 State Estimation 26

2.1.1 Linear Regression 27

2.1.2 Weighted Least Squares 28

2.1.3 Recursive Least Squares 30

2.1.4 The Kalman Filter 34

2.2 State Spaces For Visual Tracking 38

2.3 Feature Detection and Data Association 39

3 Image Registration **43**

3.1 Registration of Scalar Images 45

3.2 Constraints on Terms of Cost Function 48

3.3 Registration of Vector Images 49

3.4 Warps 51

3.4.1 Translation 52

3.4.2 Translation, Rotation, and Scaling 52

3.4.3 Affine 53

3.4.4 Piecewise Affine Warps 54

3.4.5 Piecewise Affine Registration 56

3.5 Shape Priors 58

4 Tracking Dynamic Regions of Texture and Shape **63**

4.1 Model Description 65

4.1.1 Reparametrization of the shape-free context 69

4.2 Measurement Model 70

4.3 Prediction 71

4.4 Detection 72

4.5 Update 74

4.6 Occlusion Reasoning 76

4.7 Algorithm 79

4.8 Summary 79

5 Experimental Evaluation **83**

5.1 Hurricane Tracking 83

5.2 Tracking with PTZ Cameras 85

5.3 Vehicle Tracking through Occlusion 89

5.4 Seismic Imagery 92

5.4.1 Texture Basis 95

5.4.2 Horizon Tracking 98

5.4.3 Applications 102

5.5 Group Tracking 107
5.5.1 Far-field tracking 109
5.5.2 Mid-field Tracking 113
5.5.3 Group Tracking Through Occlusion 113
5.6 Application: Image Denoising 118
5.7 Automatic Initialization of Tracks 119
5.8 Stability 121
5.9 Time Complexity 122
5.10 Assessment 124

6 Conclusion **129**
6.1 Future Work 130
6.1.1 Estimating Uncertainty 130
6.1.2 Relation to Background Modeling 131
6.1.3 Initialization 132
6.1.4 Shape Parametrization 132
6.1.5 Shape Priors 132

A Derivation of Unique Affine Transform **135**

List of Figures

1-1 Image of a car as it travels down a road. 12

1-2 The changing shape and appearance of hurricanes 13

1-3 Tracking examples 14

1-4 Blob-based and Model-based tracking 15

2-1 Measurements are inherently noisy 26

2-2 Least Squares vs. Kalman State Estimation 34

3-1 Registration example 45

3-2 Survey of warp functions 51

3-3 Piecewise-affine warps 56

3-4 Jacobian of piecewise-affine warp function 57

3-5 Spring model shape prior 61

4-1 Model-based tracking overview 65

4-2 Predicted shape and appearance of a region 71

4-3 Registration error as a function of iteration 72

4-4 Updating the model 75

4-5 Two methods of occlusion reasoning 76

5-1 Hurricane tracking in the Gulf of Mexico 85

5-2 Tracking group formations within a panning, tilting, and zooming camera. 87

5-3 Tracking group formations in the presence of two independent motion fields 88

5-4 Vehicle tracking through occlusion 91

5-5 Acquisition of seismic imagery 93

5-6 Two slices through the seismic volume 94
5-7 Perceived motion of the seismic volume 94
5-8 Pyramidal decomposition of the seismic imagery 96
5-9 Region used to learn texture subspace. 96
5-10 Sorted list of eigenvalues 97
5-11 Projection of steerable pyramid analysis onto largest three eigenvectors of PCA subspace. 98
5-12 Tracking with intensity vs. learned texture basis 99
5-13 Horizon tracking 101
5-14 Visualization of tracks in vertical cross-sections 103
5-15 Synthesizing the appearance of the horizon region 105
5-16 Compensating for the deformation of the layers reveals an interesting landscape of geologic anomalies 107
5-17 Anomaly detection using accumulation maps 108
5-18 The challenge of tracking in dense scenes 109
5-19 Traditional approaches to tracking in such environments fail 111
5-20 Group tracking in densely populated scenes 112
5-21 Piecewise affine meshes are more robust to error and local misregistrations than are simple affine patches 114
5-22 Tracking groups of people is more robust than tracking individuals 115
5-23 Group tracking under challenging conditions 117
5-24 Lack of articulation on the part of the shape parametrization causes this track to fail 118
5-25 Image Denoising 126
5-26 Mesh instantiated over region involving two opposing motion fields 127
5-27 Automatic Initialization of models 127
5-28 Successful tracking of a group of people as they turn a corner 127
5-29 Stability as a function of articulation and rigidity 128

Chapter 1

Introduction

The tracking of visual phenomena has always been a topic of great interest within the field of computer vision. It is a problem whose investigation over the years has produced many insights of both a biological and statistical nature and whose continued investigation is revealing new and exciting ways of thinking about the process of vision. It would not be an understatement, therefore, to consider it a problem of fundamental importance to the field.

Put simply, visual tracking refers to the area of scientific study concerned with associating an object in one image with the same object in subsequent images. For example, suppose we were interested in tracking the car shown in Figure 1-1 as it travels down the road. Over the course of ten seconds, for instance, the car will have moved a great distance and will have been imaged by our stationary, far-field video surveillance camera several times. As a consequence of the car moving, it will appear in a different place in each image. The problem of tracking, then, is the problem of finding the car in each image and, having been found, determining that it is the *same* car as the ones found in all the other images.

Though easy in principle, there are a number of factors that seek to confound the process of tracking. As objects move, even rigid ones, they change size, position, orientation, and appearance. A car travelling down a road will appear smaller as it moves away from the camera. A turning car changes its two dimensional pose, revealing different parts of the car and hiding others. Also, we may not get a full view of the object. Visual elements closer to the camera than the tracked car will occlude it, hiding part or all of the car from view.

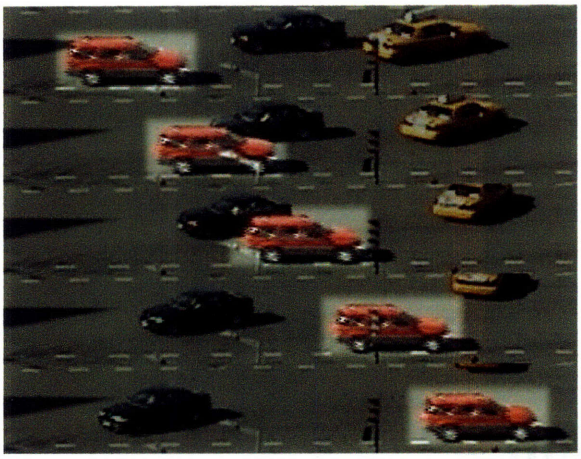


Figure 1-1: Image of a car as it travels down a road.

In addition to the challenges described above, the tracking of nonrigid objects introduces still others. Nonrigid objects can exhibit complex changes in shape that are not easily described by simple models of deformation. An example of such objects capable of undergoing complex, nonrigid deformation are meteorological phenomena such as the one shown in Figure 1-2. Hurricanes do not merely change their shape, however. Their textured appearance changes as well, further complicating the problem of tracking. A tracker that can operate effectively on such data needs to take these issues into consideration.

In this dissertation, we consider the problem of tracking objects that exhibit complex, nonuniform, nonrepeating changes to both their shape and textured appearance. By nonuniform, we mean that the textured appearance and shape of the object can change differently at one point of the object than at another. By nonrepeating, we mean that the shape and textured appearance of an object can undergo a sequence of changes that will not bring it back to any of its previous shapes or textures. We develop a model capable of representing such regions and show how it can be used to track diverse phenomena under real world conditions, including multiple and prolonged occlusion, clutter, lighting effects, and visual distractors. We demonstrate the utility and versatility of the approach by tracking a diverse set of objects, including traditional objects such as cars as well as more complex visual phenomena such as hurricanes and seismic imagery. We also show how such models enable the effective tracking of groups of people in situations where tracking would otherwise

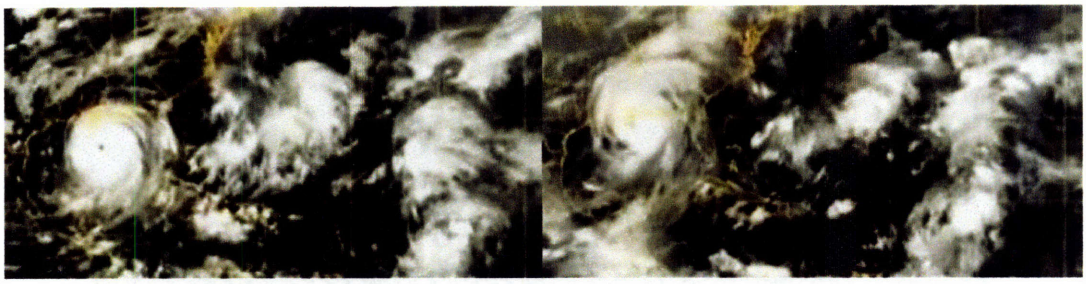


Figure 1-2: Over time, meteorological systems such as hurricanes exhibit complex changes to shape and appearance, making visual tracking of such phenomena difficult.

be difficult, or impossible. Highlights of the approach are shown in Figure 1-3.

1.1 Why Track?

There are many reasons why researchers are interested in the problem of visual tracking. Tracks – the correspondences of objects across time – are useful in a number of applications. For instance, the video surveillance and monitoring (VSAM) community are interested in, among other things, the detection of unusual or suspicious events [64, 69, 37, 33, 28]. These events can be detected by maintaining a model of common activity, derived from analyzing the output of tracking systems, in order to detect unusual activity.

Others have used tracking data for automatic ground plane rectification and camera calibration [9, 64, 45, 28, 46]. By analyzing the change in the size of objects as they move through the field of view of the camera, the perspective distortion of the ground plane can be estimated, and the scene rectified. This facilitates such tasks as identification, since the size, velocity, and other characteristic features of objects can be compared, regardless of the distance each object is from the camera, or from each other.

Tracking is also commonly used in object identification and classification [10, 69, 28, 62, 60, 34]. Many distinctive features of objects can be derived from tracking data, such as object trajectories, position, size, and appearance. These features can be fed into a classifier for automatically identifying cars and pedestrians, for example. They have also be used successfully in the automatic discovery of object classes through the use of clustering [10, 69, 34] and other unsupervised learning methods [28, 60].

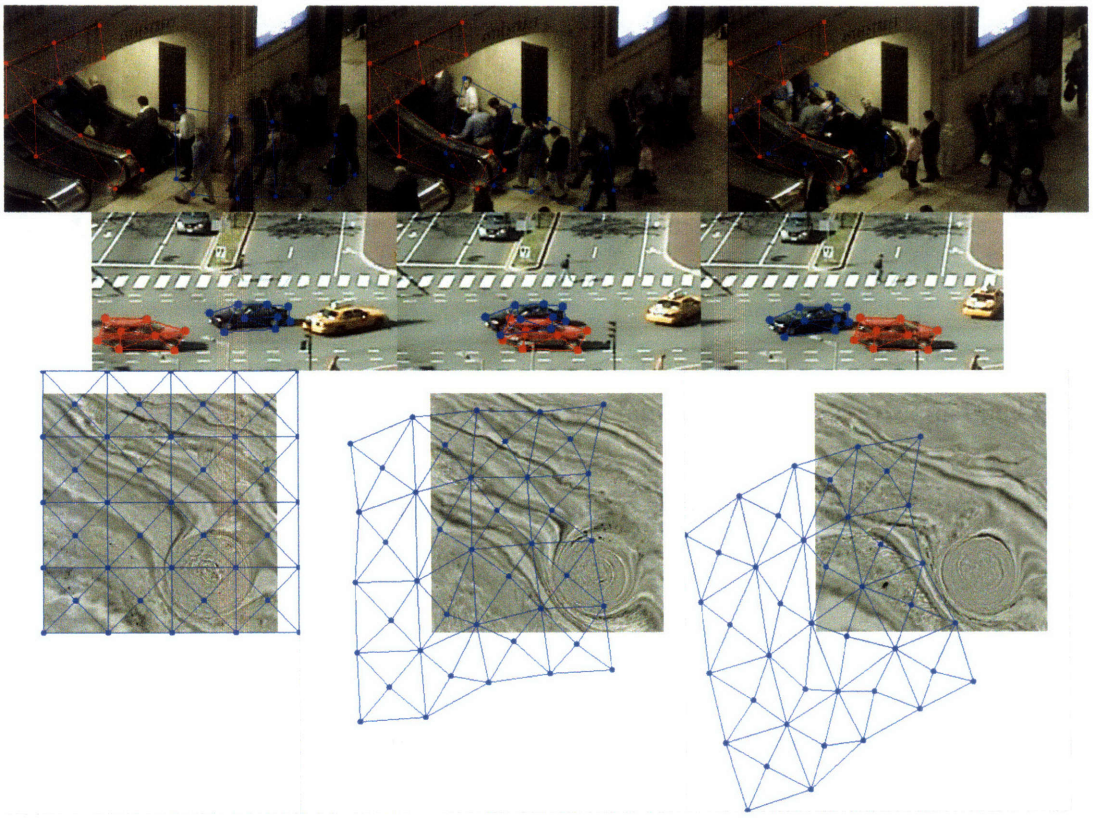


Figure 1-3: Several examples of regions tracked by the method suggested here. Such examples illustrate the ability of the method to track dynamic regions of texture and shape through occlusion, clutter, and other visual distractions. Top: group tracking in busy indoor environments. Middle: vehicle tracking in a far-field surveillance setting. Bottom: tracking deformations in soil layers in geological seismic imagery.

Another use of the correspondences established during the tracking process is for image denoising and resolution enhancement [11, 12, 3]. Imaging an object several times yields several different images of the same object. These images can be combined into a single image of the object that has more detail and color depth, and less noise, than any of the individual images themselves.

In the next section, we give a brief overview of the methods of tracking commonly employed and some of the open challenges in visual tracking.

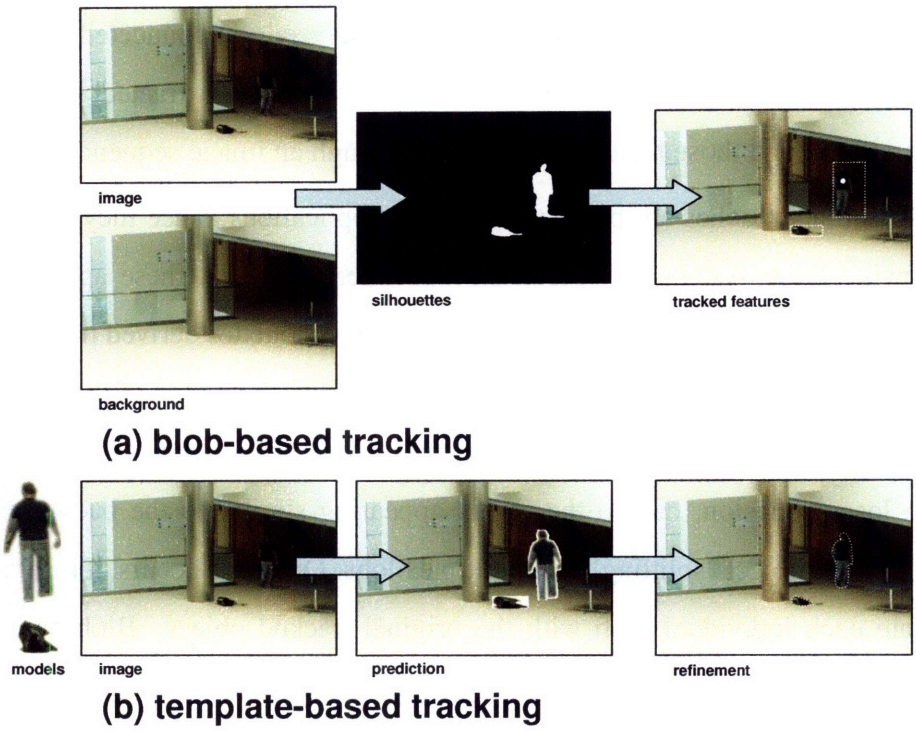


Figure 1-4: Two popular tracking paradigms include (a) bottom-up blob-based tracking and (b) top-down, template- or model-based tracking (b). In blob-based tracking, a model of the stationary background is maintained and each new image is compared to it. A binary foreground map is created consisting of those pixels in the image which differ in appearance from the background. The individual foreground pixels are then clustered into a number of independent blobs, from which salient features are extracted and tracked. In model-based tracking, explicit models of the tracked objects are maintained and a search is performed to locate them within each new image. The position, shape, and appearance of each model is first predicted within the image, according to the tracked model. A local search is then performed in order to refine the predicted appearance and achieve a good fit of each model to available imagery.

1.2 Tracking Paradigms

Though there are a myriad of tracking algorithms, they tend to fall into two broad categories: blob-based trackers and template-based trackers. Figure 1-4 shows an example of tracking under both paradigms.

Blob-based trackers follow a bottom-up methodology in which little is assumed about the nature of the objects to be tracked. For example, the shape, size, or color of the object need not be known beforehand. These approaches typically rely on the output of background subtraction in order to produce silhouettes, or binary segmentation maps

[52, 55, 21, 67, 63, 73]. The silhouettes are then analyzed via connected components in order to produce blobs. In turn, certain features of the blobs are extracted, such as centroid position and area, and those features are tracked through an image sequence.

Blob-based approaches are popular in VSAM applications, since they tend to be fast, often allowing these systems to operate in real-time. However, these approaches suffer from a number of limitations. Since they often rely on silhouettes derived from background subtraction, they are subject to the problems associated with it. One of the most common is the problem of disjoint and broken silhouettes. This arises when the tracked object's silhouette loses cohesion, due to the object appearing similar to the background. This results in a single object "splitting" into multiple blobs. Once this occurs, it is often difficult to associate each individual blob with the original tracked object. Blob-based trackers also suffer under the converse problem of "merged" silhouettes, in which two objects in projected, if not physical, proximity have their silhouettes joined. Thus, multiple tracked objects appear as one, which results in a similar problem of disambiguation.

Template-, or model-based approaches, on the other hand, follow a top-down methodology in which constraints are imposed on the appearance, shape, or other visual characteristics and features of the objects to be tracked. The use of the word "template" betrays the fact that the most widely used and successfully applied models of objects for tracking are those using simple visual templates in the form of image patches [48, 58, 29, 50, 14, 36]. These are not the only model used however. Other models of objects include those representing contours [6, 7, 32, 13, 54, 4], color histograms [53, 15, 30, 31], and subspace models [16, 20, 57, 66].

The tracking of templates, features or, more generically, models gives these types of approaches robustness to environmental effects such as changes in illumination, as well as giving them more tolerance to bad or missing visual data. The main problems associated with this type of approach are problems in initialization and model update. Initialization is often done manually, limiting the practical use of trackers based on these approaches. Furthermore, it is often unclear how and when to update the model. This could lead over time to the problem of model drift, in which the model deviates significantly in position, appearance, or shape from the actual imagery resulting, in the best case, to a lost track and

in the worst case, to an incorrect track.

There are some challenges that are universal to tracking and affect both bottom-up, blob-based trackers and top-down, model-based trackers equally. One such problem is that of occlusion, in which the object to be tracked is simply not present in the imagery, having been hidden by some other feature in the scene, such as a tree, building, or another tracked object. A related problem is that of partial-occlusion, in which the tracked object is partially hidden by some other feature, such as low-lying shrubbery, a fence, or an escalator. This presents a different challenge, because often such imagery is easily tracked, though in the case of a blob-based tracker, may result in a blob whose salient features, such as centroid, bounding box, and area will appear significantly different. The issue of partial occlusion also affects template-based trackers, since the physical appearance of the object will likewise appear significantly altered. Without also being able to detect the occluding object, it is ambiguous as to whether the object is literally changing size and appearance or whether part of it is simply being obstructed by some other element in the scene.

In the next section, we will give an overview of our method of tracking, describe how it addresses some of these open challenges, and detail the contributions it makes to the fields of tracking and computer vision.

1.3 Overview of Thesis Work

We are interested in the construction and maintenance of a model capable of compactly representing a region of imagery undergoing complex, nonuniform, nonrepeating changes to both its textured appearance and shape. We are careful not to use the term "object" to describe such phenomena since, in the course of this dissertation, we will show how to move away from the concept of individual object tracking to the more general concept of region tracking. We will show that it is possible to model jointly the dynamics and appearance of a cluster, or group, of objects and model the group and the individual motions of its members as a single region undergoing consistent, though not necessarily repeating, changes to shape and texture.

In this regard, the work described in this dissertation relates to model-based tracking.

We introduce a dynamic, statistical model of visual appearance that takes into account the complex changes to appearance and shape that regions in motion can undergo. The model consists of a joint space of appearance and shape parameters describing a dynamic region of imagery. These parameters comprise the state of a linear dynamical system and are estimated recursively, in an online model of computation where the data arrive sequentially, one at a time.

In addition to the representation of shape and texture, we describe how such a model can be used as the basis for a tracker. We describe methods of predicting shape and appearance, detecting instances of the tracked region within the imagery, and of updating the model to take into consideration the differences between the predicted appearance and the detected one.

The full tracker can be summarized as follows. At each time t, the shape and appearance of the tracked region is predicted from the model. A local search is performed, to find an instance of the model within the available imagery I_t. This search is performed in a principled, registration-based framework consisting of a gradient-descent method over parameters of a space of allowed deformations. The idea is to register I_t with the probabilistic model of shape and appearance predicted by the dynamical system. The imagery I_t need not be the raw intensity or even color values but can represent the imagery filtered through an arbitrary filter bank. We show how such filters can be integrated into the tracking and registration framework.

The ability to reason about occlusion and occlusion boundaries is fundamental to the formulation of the model described in this dissertation. Due to the generative nature of these probabilistic models, occlusion reasoning becomes a problem in likelihood estimation and classification. For each area in which two or more tracked regions overlap, the likelihood of observing the pixels within that area in I_t are evaluated under each model. For example, suppose we are tracking the two cars shown in Figure 1-3, middle. Within the area of overlap of the two cars, the red pixels would have a low likelihood when evaluated under the model of the further of the two cars and vice-versa for the black pixels.

Two methods of occlusion reasoning are then employed. The first is a per-pixel maximum likelihood estimate, in which each pixel is assumed to be a sample drawn from the

model under which it is most probable. The other is a majority-vote scheme, in which the entire area of overlap is constrained to arise from only one model. In such a manner, occlusion masks are likewise predicted along with the shape and appearance of each tracked region. This information is taken into consideration during the registration process, which fits the model to the available imagery.

With an instance of the tracked region obtained by registering model and imagery, and the visibility of each pixel within that region estimated by the process described above, the tracked model of shape and appearance can be updated. It is updated by integrating the difference between the shape and appearance of the detected region and that predicted by the model into a new estimate of the shape and appearance parameters of the tracked region.

With this new information suitably integrated, the processes of prediction, detection, occlusion reasoning, and reestimation of shape and appearance parameters can continue for the next time step. Because this is a recursive process, relying only on the models maintained up until the last time step, it can be repeated to track for arbitrary t.

1.3.1 Algorithm Parameters

The work described in this dissertation includes a representation of shape and textured appearance and a method of using such a representation as the basis for an online visual tracking system. Such a system requires certain information to be known before tracking can begin, and requires other information on a regular basis. The resultant tracks and all intermediate products, such as the detected regions, form the output of the system. From an algorithmic perspective, the requirements on such a tracker and the usable output of it consist of:

Given A set of m models $M_{i,0}, i = 1...m$ initialized at time $t = 0$. These consist of the initial shape and appearance of each region to be tracked and constitute the initial states of the dynamical systems.

Inputs At each time t, a new multidimensional image I_t.

Outputs At each time t, the models $M_{i,t}$ consisting of distributions over shape and texture of the tracked regions of imagery. Also, the best fit of models to imagery which comprise the detected regions, and their occlusion masks, indicating which parts of the regions are currently visible and present within I_t.

1.3.2 Applications

The models maintained during the process of tracking visual phenomena are generative probabilistic models with numerous applications. Some of the applications explored in this dissertation are described below.

Detection A method is described for detecting the presence or absence of a model within the imagery. During tracking, this method is used to locate instances of each model within the imagery.

Synthesis We describe how a learned model can be used to predict the appearance of an object or region when part or all of it is hidden from view.

Image Denoising We show how the model itself as well as its instances in time can be used to enhance the image of the tracked object or region.

Segmentation We describe a procedure for automatically separating overlapping regions of imagery whose appearance and shape deformations are distinct and separable.

Anomaly Detection By treating the estimated probabilistic model of shape and appearance as "usual", anomalous regions are detected as outliers from such a model. Results are shown on seismic imagery and are verified by comparison with manual segmentations performed by a geologist.

1.3.3 Evaluation

We evaluate the tracker based on the appearance model proposed here in several ways. First and most importantly, we track a diverse set of visual phenomena under a range of

different conditions. Some of those conditions include indoor and outdoor scenes, near-field and far-field camera positions, tracking through clutter and other unexplained visual distractors, and tracking through sustained occlusion. We track rigid and nonrigid objects alike, and show several examples of how the model can be used to enable the tracking of groups of people.

To further evaluate the accuracy of the tracker, we use the resultant tracks and the model estimated at each time t in several applications, described above. The accuracy and fidelity of these applications are direct indicators of the accuracy of the tracking process itself. In particular, the accuracy of the system is independently verified by validating the results of anomaly detection within the seismic imagery against data labeled by an expert geologist. The results indicate that all of the annotated data were detected within the seismic imagery and also suggest that there are many other such anomalies present that have not yet been considered.

To evaluate the stability of the approach, we track a region multiple times while systematically altering two important parameters of the model: the degree of articulation, and the rigidity of the shape. The results indicate that, as the degree of articulation increases, so too does the deformation of the shape. The converse is true for rigidity: as the shape is made more rigid, its deformations become less pronounced over the course of the tracked sequence. Thus, the terms have inverse effects on the shape of the tracked region and the results indicate that there is a range of values for model articulation and rigidity that produce compelling results.

1.4 Contributions

In this section, we highlight the main contribution this thesis makes to the field of visual tracking.

1.4.1 Joint model of shape and appearance

We introduce a dynamic model of shape and appearance that is capable of compactly representing the appearance of a region of imagery as it undergoes changes to both its textured

appearance and shape. Similar models, such as active appearance models [16] and active blobs [57] represent shape and appearance jointly, in low dimensional linear subspaces. These lead to compact object models, but are difficult to use for general tracking. This is because, in the case of active appearance models, labeled training data is required to construct such models and, in both cases, the models of the objects to be tracked need to be known before tracking begins. The model of shape and appearance described herein requires no labeled training data and is maintained in an online estimation and tracking framework, thus allowing the tracking of novel objects that have not yet been observed.

Similar models of texture include those which model the change in textured appearance of a region as a linear dynamic on appearance coefficients, referred to as dynamic textures [18]. These models are suitable for representing textured regions that exhibit certain properties of stationarity, such as a localized region of water waves, or smoke rising from a candle flame. Two difficulties arise from using these models for tracking, however. The first is that the phenomena need to be unmoving in space, which makes them suitable for background modeling [73], but not for tracking moving foreground objects. Also, in general, the appearance changes of tracked objects will not show such repeating, textured patterns, but will change in more complex and nonrepeating ways. An example of such a change is the change in appearance induced by an object undergoing a change in pose, or of a group of people in which the constituents of the group change positions.

Fitzgibbon alleviates the problem of modeling moving textures in [22] by decomposing the change in appearance of a scene into a global, parametric motion component and a local, stochastic component modeled similarly to dynamic textures. By estimating the global motion of the scene, such motions can be compensated for to provide a stabilized reference frame from which to model the more stochastic component as a dynamic texture. The global motion is modeled as translation, which is suitable for modeling the panning or tilting motion of a PTZ camera, but is not suitable for modeling more complex changes to the shape of an object, such as those caused by pose changes or changes in perspective.

Other models are similar in that they estimate parameters of appearance and shape in an online tracking framework [29, 36, 39]. These approaches focus on modeling specific modes of appearance variation, for example those caused by illumination changes or mod-

eling sudden vs. gradual changes in appearance. The shape models used are similarity transforms, suitable for modeling objects that change orientation, position, and size within the imagery, but do not undergo more complex changes, such as changes in articulation or pose.

As such, the approaches described above would all have difficulty modeling the shape and appearance changes that the regions we seek to track can undergo. These include the shape and appearance changes involved in modeling groups of people, where such changes are often nonuniform and nonrepeating in nature and whose variability in shape and appearance over all time do not cluster in the joint space over such parameters. It also includes the modeling of shape and texture changes of more abstract data, such as seismic imagery that can deform and change appearance in complex ways unrelated to traditional sources of appearance variation, such as those caused by changes in illumination.

1.4.2 Group Tracking

By generalizing the concept of object tracking to that of region tracking, we show how to model the complex motion of groups of people and of the individuals within those groups. In this way, we are able to track under conditions where other trackers would fail due to the complexity of the scene and the complexity of the interactions within the region.

1.4.3 Occlusion Reasoning

We describe how the appearance model described in this dissertation leads to simple and efficient online occlusion reasoning. By considering each model as a generative process within a larger probabilistic framework, we treat the problem of occlusion reasoning as a problem in classification. We then show how even simple classifiers can be used to effectively estimate an ordering of the appearance models. Furthermore, by reducing it to a conventional problem in classification, we show how more sophisticated classifiers can be seamlessly integrated into the tracker to achieve more robust and accurate occlusion reasoning.

1.5 Outline of Dissertation

The remainder of this dissertation will proceed as follows. Chapter 2 gives an overview of various methods of tracking visual phenomena and describes the Kalman filter as it relates to the statistical model of shape and appearance suggested here. Chapter 3 describes image registration as it relates to model-based visual tracking and the extensions to it that are derived for this work. In chapter 4, we describe in detail the model of shape and appearance introduced herein and describe how it can be used for tracking in an online estimation framework. Chapter 5 demonstrates the utility of the model in tracking visual phenomena as well as illustrating how it can be used in a number of applications. We conclude in chapter 6 with a discussion of the limitations of the current approach and a discussion about possible avenues of future investigation.

Chapter 2

Tracking

In Chapter 1, we referred to tracking as the process of associating an object in one image with the same object in subsequent images. While this is certainly true, it does not tell the whole story. Hidden in that description is the challenge of *detecting* the object in the images.

The process of detection can take many forms. For example, "detecting" the car may mean finding its position, or the position of its centroid. It may also mean finding its outline, or shape. In this way, the process of detecting an object is the process of extracting features of that object from the imagery. For now, we will use the word "detect" to describe any method of finding the object or features of the object within the imagery without implying anything about those features.

The process of detecting an object is often correlated with the task of association. For example, it is often easier to detect an object at time t if we have already detected it at time $t-1$. Using the example shown in Figure 1-1 of the car travelling down the road, knowing where the car was, or what it looked like, at a previous time constrains where it can be (or what it can look like) at the present. Certainly it will not have travelled very far from where it was previously, nor will it look significantly different. Thus, knowing its position and appearance previously can tell us a lot about its position and appearance now.

However, though the car may not have moved very much, we would not expect that its position now will be the same as its position then. If we are tracking the centroid of the car, for example, we would not expect its location to be the same at time $t-1$ as it is at time t.

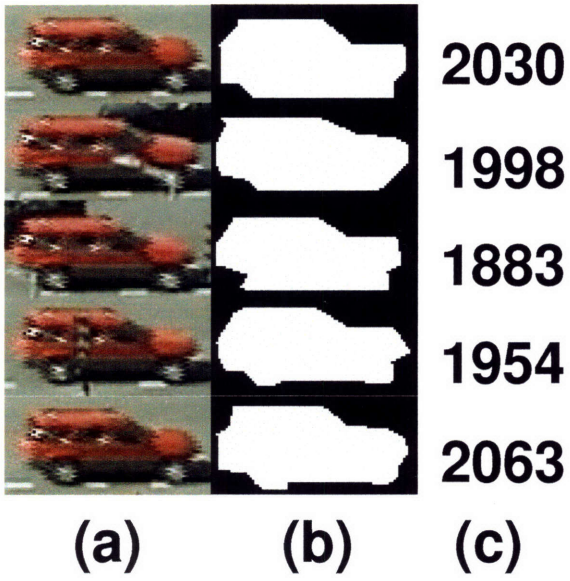


Figure 2-1: The determination of object sizes is an inherently noisy process. Even manual segmentations (column b) of objects show a range of values for object size (column c). The size of the object is computed by counting the area, in pixels, taken up by the segmentation.

It would have moved by some $\Delta u = (\Delta x, \Delta y)^T$ in relation to its velocity and the time Δt between images. If we could estimate the change in position of the car is it travels down the road, it may further facilitate the tasks of detection and association.

Thus the field of visual tracking is primarily concerned with the estimation or determination of three quantities: the extraction of object features from imagery (e.g. shape, centroid, etc.), the association of feature sets with tracked objects, and the maintenance and estimation of the *state* of tracked objects (e.g. position, velocity, shape, appearance, etc). Since state estimation plays such a significant role in the tasks of detection and association, we begin our discussion of visual tracking there. Much of the following section is derived from, and a unification of, the excellent introductions to the topic by Strang [65] and Welch and Bishop [70].

2.1 State Estimation

Classically, state estimation refers to the problem of separating signal from noise in a recursive framework suitable for online processing. The state is the unobserved, "pure" signal

from which we sample noisy measurements. The term "state" refers to the state of a dynamical system which is used to estimate such a signal. The relevance of this will become apparent in Section 2.1.4.

For now, suppose we wish to measure a quantity, but our measurements are corrupted by noise. This scenario happens all the time: we measure a heartbeat three times in a row and get three close, but different answers – 69, 72, 70. How fast, then, is the heart really beating? Or, suppose we wish to know how hot it is outside, but three thermometers read three different temperatures – $70°F$, $70.6°F$, and $71°F$. Or we wish to know how much we weigh – 150lbs, 149lbs, and 153lbs.

Similar types of questions are asked in vision applications. For example, suppose we wish to estimate the size of an object, such as the car in Figure 1-1. If we can extract its silhouette, the size of the car can be determined by counting the number of pixels covering it. However, we may get a different count at each time t. Figure 2-1 shows the extracted silhouettes of the car and associated measurements of size, in pixels – 2030, 1998, 1883, 1954, and 2063. What, then, is its "true" size? We may also wish to estimate its velocity v, based on estimates of centroid position $p = (x, y)^T$ and interframe time differences Δt:

$$v_t = (p_t - p_{t-1})\Delta_t^{-1}. \tag{2.1}$$

The estimation of state, therefore, is the estimation of these true quantities – the size of the car, the number of heartbeats, etc. – based on noisy measurements, or observations. The simplest and most intuitive method of arriving at such estimates is through the use of least squares linear regression.

2.1.1 Linear Regression

Linear regression assumes a (known) linear relationship between the unobserved state x and the observed measurements $z = (z_0, z_1, z_2, ..., z_t)^T$ up to time t:

$$\mathbf{A}x = z. \tag{2.2}$$

The matrix $\mathbf{A}$ relates x to all observations z_i and is constructed by:

$$\mathbf{A} = \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \\ \cdots \\ \mathbf{H}_t \end{bmatrix}, \tag{2.3}$$

where each $\mathbf{H}_i$ relates the state x to the measurements z_i taken at time i. Typically, all $\mathbf{H}_i$ are equal, but they do not have to be.

For example, suppose we wish to determine the size of the car shown in Figure 2-1 from its silhouettes extracted from the imagery at different times t_i. In this situation, the state is the size $x = s$, the measurements are the pixel counts at different times $z_i = c_i$, and the relation between each measurement and the state vector is $\mathbf{H} = \mathbf{I}_1$. Each row of $\mathbf{A}$ is the identity relation.

When the number of observations equals the size of the state vector, the system is exactly determined. When estimating the size of the car from one silhouette, the best estimate is the size of that silhouette. However, the number of observations often exceeds the quantity being determined. We get a silhouette at each t_i. The system is overdetermined, and there is no exact solution. The best estimate, in a least-squares sense, comes from the normal equations:

$$\begin{aligned} \mathbf{A}^T\mathbf{A}\hat{x} &= \mathbf{A}^T z \\ \hat{x} &= (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T z, \end{aligned} \tag{2.4}$$

which gives the estimate $\hat{x}$ of true state x that minimizes the error $||e(x')|| = ||z - \mathbf{A}x'||$.

2.1.2 Weighted Least Squares

It is often the case that we can trust some measurements more than we can trust others. One scale is more accurate, so its measurement of weight can be trusted more. Or, one silhouette is particularly clean, so the estimate of the car's size derived from it can be trusted more. Also, measurements of different quantities may have inherently different accuracies. In the estimation of the car's velocity, for example, the interframe time differences Δt_i are very

accurate, whereas the estimates of centroid position, which are based on silhouettes, may be decidedly less so. The method of weighted least squares takes into consideration that some measurements can be trusted more than others. It does this through the inclusion of a weight matrix W in Equation 2.2:

$$\mathbf{WA}x = \mathbf{W}z. \tag{2.5}$$

The error to be minimized in this case is $||We|| = ||\mathbf{W}z - \mathbf{WA}x||$, which leads to the following set of equations:

$$\begin{aligned} (\mathbf{WA})^T\mathbf{WA}\hat{x} &= (\mathbf{WA})^T\mathbf{W}z \\ \mathbf{A}^T\mathbf{W}^T\mathbf{WA}\hat{x} &= \mathbf{A}^T\mathbf{W}^T\mathbf{W}z \\ \hat{x} &= (\mathbf{A}^T\mathbf{CA})^{-1}\mathbf{A}^T\mathbf{C}z, \end{aligned} \tag{2.6}$$

where $C = \mathbf{W}^T\mathbf{W}$, a symmetric positive semi-definite matrix.

The equations above are guaranteed to find the estimate $\hat{x}$ that will minimize $||\mathbf{W}e||$ for a given $\mathbf{W}$. But which $\mathbf{W}$ should be chosen? With $\mathbf{W} = \mathbf{I}$, we are back to ordinary least squares, and so have not taken advantage of the fact that some measurements are more precise than others. Intuitively, we would like the errors to mean more when the measurement can be trusted more. Said another way, if $\mathbf{W}$ were diagonal, so the weights associated with each measurement are all independent, then we would want w_i to be greater when z_i could be trusted more.

The answer lies in a statistical interpretation of the errors. We do not know what each e_i is, otherwise we could compensate for it in our model. But we can make some assumptions about the population that gives rise to such errors. Let v_i be a random variable from which e_i is one value. Let v_i have 0-mean so that $E[v_i] = 0$. This is not so onerous, however, since if $E[v_i] = c$, with a change of variable $v_i' = v_i - c$, we can always write it as $E[v_i] = E[v_i' + c] = E[v_i'] + c$. Thus, we can simply remove the constant c from the random variable to arrive at a 0-centered one. Furthermore, let $E[v_i^2] = \sigma_i^2$. The variance is a measure of spread of the errors in the measurement and, hence, is an indicator of its precision. The smaller σ_i is, the more we can trust z_i. Intuitively then, we would like

$w_i = \sigma^{-1}$.

For diagonal $\mathbf{W}$, this is the correct choice. For general $\mathbf{W}$, it is close. Let $\mathbf{V} = E[vv^T]$ be the covariance matrix of the errors for all observations. The correct choice, then, is when $\mathbf{W}^T\mathbf{W} = \mathbf{C} = \mathbf{V}^{-1}$.

The correct choice with respect to what quantity though? If we were just trying to minimize prediction error, $\mathbf{W} = 0$ would be the best choice. But then $\hat{x}$ would look nothing like the true state x. The goal is to estimate x, for example the size of the car, with as much accuracy as possible, not necessarily to minimize prediction error. We wish to set $\mathbf{W}$ such that $\hat{x}$ looks as much like x as possible. But we do not know x so the best we can do is minimize the expected error between it and our estimate. Let $\mathbf{L} = (\mathbf{A}^T\mathbf{C}\mathbf{A})^{-1}\mathbf{A}^T\mathbf{C}$. According to Equation 2.6, $\hat{x} = \mathbf{L}z$. It is straightforward, then, to show that $E[x - \hat{x}] = 0$ whenever $\mathbf{L}\mathbf{A} = \mathbf{I}$. This means that our estimate $\hat{x}$ will be unbiased, regardless of choice of $\mathbf{C}$. Our estimates of the size of the car will hover around its true size. The quantity we wish to minimize, then, is the expected variation in estimates about the true state: $E[(x - \hat{x})(x - \hat{x})^T]$. This is guaranteed to be smallest when $\mathbf{C} = \mathbf{V}^{-1}$. The estimates of true state will be best when the regression is weighted by the inverse of the covariance of the measurement uncertainty. We can arrive at the tightest estimate of the car's size when we weigh the measurements in relation to the expected error in those measurements. The covariance $\mathbf{P}$ of the optimal estimate $\hat{x}$ is thus related to the covariance $\mathbf{V}$ of the measurement noise according to:

$$\mathbf{P} = (\mathbf{A}^T\mathbf{V}^{-1}\mathbf{A})^{-1}. \tag{2.7}$$

For a more detailed analysis, we refer the reader to [65].

2.1.3 Recursive Least Squares

The previous section ended with the derivation of the covariance $\mathbf{P}$ of the optimal estimate $\hat{x}$ of the true, but unknown and unobservable, state x. It is rarely used, however, because it estimates x in batch. All observations across time are needed to establish the estimate. For an approach to be practical, however, it needs to be able to operate online – as the

observations come in. It is simple enough to use weighted least squares in this manner: just keep adding rows to $\mathbf{A}$. But this quickly becomes intractable. A better way would be to use the optimal estimate $\hat{x}_{t-1}$ at time $t-1$ and improve upon it, given new observations at time t, in order to arrive at the optimal estimate $\hat{x}_t$ at time t. But the resulting $\hat{x}$ must be the same, *as if* we were using all observations up until time t to arrive at the estimate. It must produce the same result as the weighted least squares estimate would produce using all the data.

When dealing with an online, or recursive, approach, it is helpful to pose the problem in those terms. Recall that the matrix $\mathbf{A}$ that relates state to observations is $\mathbf{A} = [\mathbf{H}_1^T, \mathbf{H}_2^T, ...\mathbf{H}_t^T]^T$. Each new set of observations adds another block $\mathbf{H}_i$ to $\mathbf{A}$. Let us assume all $\mathbf{H}_i$ are the same, which implies that we get the same types of observations at each time step. For example, at each time step, we compute object size and centroid position. Let us further assume that the measurement errors v_t are independent of prior errors and that their covariance does not change with time. Thus, $\mathbf{V}_t = \mathbf{V}$.

The problem can now be posed as follows: given state estimate and covariance $\hat{x}_0$, $\mathbf{P}_0$ at $t = 0$, and measurement z_1 at time $t = 1$, calculate the optimal $\hat{x}_1$ and $\mathbf{P}_1$ without using z_0. A solution that does not depend on the previous measurement, or any other measurement except the current one, may be applied recursively and, therefore, in an online system that gets measurements one by one, as they occur.

We know from the solution to weighted least squares that the optimal estimate $\hat{x}_1$ and covariance $\mathbf{P}_1$ for the complete system involving both measurements z_0 and z_1 is

$$\begin{aligned} \mathbf{A}^T\mathbf{C}\mathbf{A}\hat{x}_1 &= \mathbf{A}^T\mathbf{C}z \\ \mathbf{P}_1 &= (\mathbf{A}^T\mathbf{C}\mathbf{A})^{-1}. \end{aligned}$$

In this case, we have

$$\begin{aligned}
\mathbf{A} &= \begin{bmatrix} \mathbf{H} \\ \mathbf{H} \end{bmatrix} \\
\mathbf{C} &= \begin{bmatrix} \mathbf{V}^{-1} & 0 \\ 0 & \mathbf{V}^{-1} \end{bmatrix} \\
z &= \begin{bmatrix} z_0 \\ z_1 \end{bmatrix}^T .
\end{aligned}$$

Note that it is exactly $\mathbf{P}_1^{-1}$ multiplying $\hat{x}_1$ on the left hand side of this equation. When multiplied out, we get:

$$\begin{aligned}
\mathbf{P}_1^{-1} = & \begin{bmatrix} \mathbf{H} \\ \mathbf{H} \end{bmatrix}^T \begin{bmatrix} \mathbf{V}^{-1} & 0 \\ 0 & \mathbf{V}^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{H} \\ \mathbf{H} \end{bmatrix} \\
\mathbf{P}_1^{-1} = & \ \mathbf{H}^T\mathbf{V}^{-1}\mathbf{H} + \mathbf{H}^T\mathbf{V}^{-1}\mathbf{H}
\end{aligned} \tag{2.8}$$

Since $\mathbf{P}_0^{-1} = \mathbf{H}^T\mathbf{V}^{-1}\mathbf{H}$, this reduces to

$$\mathbf{P}_1^{-1} = \mathbf{P}_0^{-1} + \mathbf{H}^T\mathbf{V}^{-1}\mathbf{H}. \tag{2.9}$$

Previously, the covariance of the optimal state estimate was described in terms of the full observation matrix $\mathbf{A}$ and the measurement covariance for *all* observations. It is now described recursively, using only its previous setting $\mathbf{P}_0$ and the *current* measurement covariance $\mathbf{V}$. We can do the same for $\hat{x}_1$:

$$\begin{aligned}
\hat{x}_1 &= \mathbf{P}_1 \begin{bmatrix} \mathbf{H} \\ \mathbf{H} \end{bmatrix}^T \begin{bmatrix} \mathbf{V}^{-1} & 0 \\ 0 & \mathbf{V}^{-1} \end{bmatrix} \begin{bmatrix} z_0 \\ z_1 \end{bmatrix} \\
&= \mathbf{P}_1 \left(\mathbf{H}^T\mathbf{V}^{-1}z_0 + \mathbf{H}^T\mathbf{V}^{-1}z_1 \right).
\end{aligned}$$

Since $\mathbf{P}_0^{-1}\hat{x}_0 = \mathbf{H}^T\mathbf{V}^{-1}z_0$, we have

$$\begin{aligned}
\hat{x}_1 &= \mathbf{P}_1\left(\mathbf{P}_0^{-1}\hat{x}_0 + \mathbf{H}^T\mathbf{V}^{-1}z_1\right) \\
&= \mathbf{P}_1\left((\mathbf{P}_1^{-1} - \mathbf{H}^T\mathbf{V}^{-1}\mathbf{H})\hat{x}_0 + \mathbf{H}^T\mathbf{V}^{-1}z_1\right) \\
&= \hat{x}_0 + \mathbf{P}_1\left(\mathbf{H}^T\mathbf{V}^{-1}z_1 - \mathbf{H}^T\mathbf{V}^{-1}\mathbf{H}\hat{x}_0\right) \\
&= \hat{x}_0 + \mathbf{K}_1\left(z_1 - \mathbf{H}\hat{x}_0\right),
\end{aligned}$$

where $\mathbf{K}_1 = \mathbf{P}_1\mathbf{H}^T\mathbf{V}^{-1}$. Thus we can write the optimal estimate $\hat{x}_1$ recursively as well, using only the current measurement z_1, the current state covariance $\mathbf{P}_1$, the previous state estimate $\hat{x}_0$, and measurement covariance $\mathbf{V}$.

This result is both interesting and intuitive. Recall that $\mathbf{H}$ relates state to observation and that $\hat{x}$ is the estimate of state. $\mathbf{H}\hat{x}_0$, then, can be considered the *prediction* of the observation z_1. Thus the term $(z_1 - \mathbf{H}\hat{x}_0)$ is the *residual*, sometimes called the *innovation*, and describes the difference between prediction and measurement. If this were $\mathbf{0}$, then the prediction exactly equals measurement and we would expect that the best estimate of state at time $t = 1$ would be the same as it was at $t = 0$. When the prediction and residual are not the same, the *gain* matrix $\mathbf{K}_1$ determines how much $\hat{x}_1$ should rely on the previous estimate $\hat{x}_0$ versus the residual. It is the matrix equivalent of a ratio – the ratio, in this case, between the uncertainty in the estimate and the uncertainty in the measurements. If $\mathbf{V}$ is small, such that $\mathbf{V}^{-1}$ is large, $\mathbf{K}$ is large, and we trust the measurements more. The best estimate of state $\hat{x}_1$ is pushed more closely towards z_1. The converse is also true. When $\mathbf{V}$ is large, we do not trust the measurements, and so the estimate of state is not so easily influenced by observation.

To sum up, writing the equations in a recursive manner, we have:

$$\hat{x}_t = \hat{x}_{t-1} + \mathbf{K}_t\left(z_t - \mathbf{H}\hat{x}_{t-1}\right) \tag{2.10}$$

$$\mathbf{P}_t^{-1} = \mathbf{P}_{t-1}^{-1} + \mathbf{H}^T\mathbf{V}^{-1}\mathbf{H}, \tag{2.11}$$

with estimate of state $\hat{x}_t$ at time t and uncertainty $\mathbf{P}_t$ in that estimate, measurement z_t and

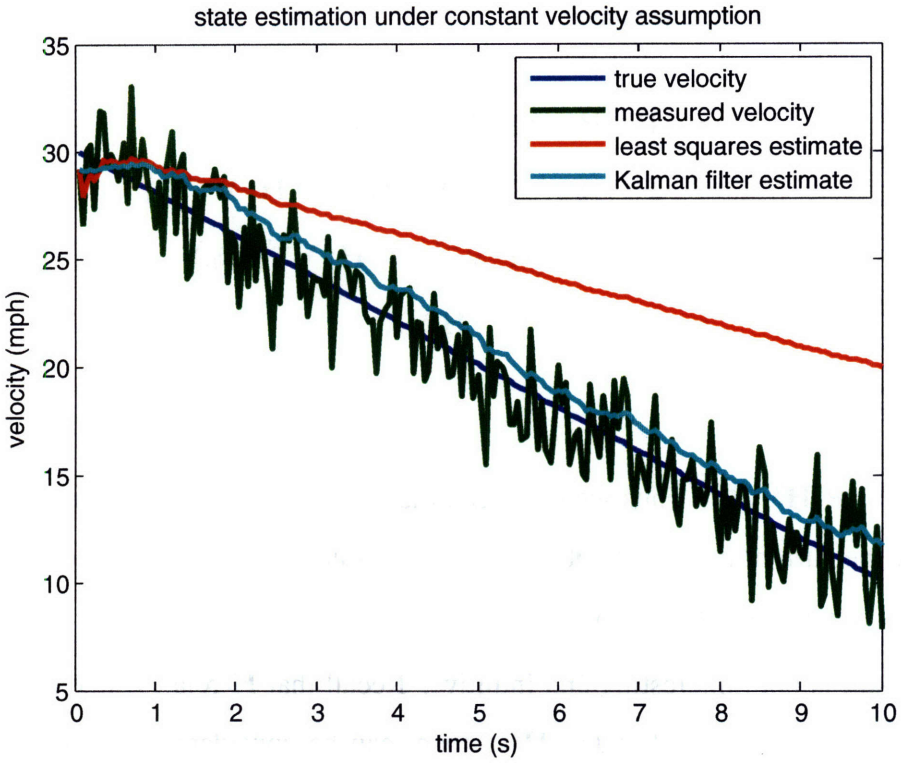


Figure 2-2: Online estimates of a linearly decreasing velocity (constant deceleration) with constant velocity assumptions. The recursive least squares estimate amounts to the average of the observed velocities up until time t. The Kalman filter estimate uses the same constant velocity assumption, but allows for a small amount of uncertainty in that dynamic. With that simple addition, the estimates of velocity are greatly improved.

measurement covariance $\mathbf{V}$, the matrix $\mathbf{H}$ relating state to measurement, and gain matrix $\mathbf{K}_t = \mathbf{P}_t\mathbf{H}^T\mathbf{V}^{-1}$.

2.1.4 The Kalman Filter

Recursive least squares works well when the quantity being estimated remains constant. The number of heartbeats per minute is not changing, the temperature is constant, or the car's velocity, as measured by centroid position vs. time, is not changing. But what if the quantity we are estimating is not standing still? We wish to estimate the car's position, but the car is moving, and thus so is the state. Can we continue to track optimally – to estimate the state of the dynamical system with minimum expected error and no bias – when the state itself is evolving? As it turns out, we can.

The Kalman filter adds two key contributions to the field of optimal recursive state estimation. First, it describes how to estimate the state of a system when the state itself changes over time: $x_t = \mathbf{A}x_{t-1}$. It also allows for uncertainty in the dynamical system: $x_t = \mathbf{A}x_{t-1} + w$, with random variable w representing the uncertainty in the dynamic $\mathbf{A}$. Thus, we are back to least squares when $\mathbf{A} = \mathbf{I}$ and $w = \mathbf{0}$.

The importance of adding uncertainty to the dynamical model should not be underestimated. To motivate this via an example, suppose we are estimating a car's velocity, and we *think* it is travelling at a constant speed. Thus, we think the dynamics of the system are $v_t = v_{t-1}$. We could be wrong, however, and almost certainly are. No car travels at a perfectly constant rate of speed, even when that is the driver's intention. Furthermore, we could be estimating the velocity of the car, under an assumption of constant velocity, when the car is speeding up or slowing down.

In a graph of velocity over time, like the one shown in Figure 2-2, the actual velocity may be decreasing linearly, as evidenced by the negatively sloping trend. The least squares estimate at each time t is the average of all measurements taken so far. If we knew, with perfect certainty, that the car was indeed travelling at a constant speed, and so all the differences in observed velocity were due to bad measurements, then taking the average is certainly the best that could be expected. The resulting estimates, however, are still not very compelling. But, if we allow that there is even a small chance that we got it wrong, that the car is, in fact, slowing down or speeding up or was otherwise not maintaining a constant speed, then taking the average of the measurements becomes far from optimal. By simply injecting a small amount of uncertainty into the model of the evolution of state, we can arrive at a much more accurate estimate of the true velocity, one that may even be qualitatively described as "good".

This is significant, as good estimates of state can be obtained without the need to know the precise dynamic under which the state evolves, which can often be quite complex. At any given time, the car may be slowing down, speeding up, or maintaining its speed. With the simple assumption of a stationary dynamic, injected with a suitable amount of uncertainty, reasonable estimates of its velocity are obtainable, even under all of these diverse conditions.

As described in [70], the mathematical model of the Kalman filter begins with a description of the state evolution and measurement processes:

$$x_t = \mathbf{A}x_{t-1} + w_{t-1} \tag{2.12}$$

$$z_t = \mathbf{H}_{t-1}x_t + v_{t-1}, \tag{2.13}$$

where $\mathbf{H}$, as before, is a matrix which describes the relation between state and measurement, $\mathbf{A}$ describes the evolution of state from $t-1$ to t, and w and v are random variables describing the uncertainty in the dynamic and the uncertainty in the measurements, respectively. Though they can, in general, change with each t, for the purposes of describing the filter we shall assume that they remain constant. They are assumed Gaussian, with covariances $\mathbf{Q}$ and $\mathbf{V}$, such that

$$\begin{aligned} w &\sim \mathcal{N}(0, \mathbf{Q}) \\ v &\sim \mathcal{N}(0, \mathbf{V}). \end{aligned}$$

The Kalman filter works by maintaining the first two moments (mean and covariance) of the distribution of state, such that

$$\begin{aligned} E[x_t] &= \hat{x}_t \\ E[(x_t - \hat{x}_t)(x_t - \hat{x}_t)^T] &= \mathbf{P}_t. \end{aligned}$$

Recall that these are the same quantities estimated by least squares. They continue to be estimated optimally, but now it is in the presence of an evolving state with uncertainty about the dynamic.

In the formulation of the filter framework, it is necessary that the noise terms w and v be independent of each other, and of x. It is also helpful to assume that they are normally

distributed. The reason for this is in the recursive nature of the filter. At each new time t and before a new measurement is associated with it, the state is driven forward, according to $\mathbf{A}$, to *predict* the state: $\hat{x}_t^- = \mathbf{A}\hat{x}_{t-1}$. But there is an uncertainty w in the prediction introduced at this time, distributed as a Gaussian with covariance $\mathbf{Q}$. Furthermore, it is an uncertainty that is added to an already Gaussian distributed state estimate, which is maintained by the filter. The latter can be represented as a random variable $X_{t-1} \sim \mathcal{N}(\hat{x}_{t-1}, \mathbf{P}_{t-1})$. The sum of these random variables at time t is $X_t = (\mathbf{A}X_{t-1} + w) \sim \mathcal{N}(\mathbf{A}\hat{x}_{t-1} + 0, \mathbf{A}\mathbf{P}_{t-1}\mathbf{A}^T + \mathbf{Q})$ and we therefore have the a-priori estimate of state:

$$\hat{x}_t^- = \mathbf{A}\hat{x}_{t-1} \tag{2.14}$$

$$\mathbf{P}_t^- = \mathbf{A}\mathbf{P}_{t-1}\mathbf{A}^T + \mathbf{Q}. \tag{2.15}$$

The prediction is an a-priori estimate at time t, because we have not yet associated a measurement with it. It has the effect of moving the estimate of state forward, according to the linear dynamic $\mathbf{A}$, and adding the uncertainty w in the dynamic model to the uncertainty in the current estimate of state. We become less certain about the true value of the state x at time t than we were at the end of $t-1$. With its incorporation into the state estimate, the remainder of the estimation process looks similar to recursive least squares. Least squares needed no such prediction step, because $\mathbf{A} = \mathbf{I}$ and $w = 0$, leaving the prediction $\hat{x}_t^-$, $\mathbf{P}_t^-$ identical to $\hat{x}_{t-1}$ and $\mathbf{P}_{t-1}$.

A new measurement z_t now comes in, and with it its own uncertainty v. This new information must be integrated into $\hat{x}_t$. It is done according to

$$\hat{x}_t = \hat{x}_t^- + \mathbf{K}_t(z_t - \mathbf{H}\hat{x}_t^-) \tag{2.16}$$

$$\mathbf{P}_t = \mathbf{P}_t^- - \mathbf{K}_t\mathbf{H}\mathbf{P}_t^-, \tag{2.17}$$

with $\mathbf{K}_t = \mathbf{P}_t^-\mathbf{H}^T(\mathbf{H}\mathbf{P}_t^-\mathbf{H}^T + \mathbf{V})^{-1}$. This result is analogous to least squares, except where Equation 2.11 references $\mathbf{P}_{t-1}$ in the recursion, Equation 2.17 references the forward

predicted covariance $\mathbf{P}_t^-$ instead. Also, the gain matrix $\mathbf{K}_t$ for least squares is most easily derived in terms of $\mathbf{P}_t$, whereas for the Kalman filter derivation, the opposite it true. $\mathbf{P}_t$ is more readily described in terms of $\mathbf{K}_t$, with $\mathbf{K}_t$ given in terms of $\mathbf{P}_t^-$. The interpretation of $\mathbf{K}_t$ remains the same, however, determining how much $\hat{x}_t$ should be influenced by the prediction $\hat{x}_t^-$ and how much it should be influenced by the new measurement z_t.

2.2 State Spaces For Visual Tracking

A description of the representation, or state space, comprises an integral part of any tracker [36, 35]. It is the value or values that will be estimated over the course of a tracked sequence. We describe representations that are most similar to the one suggested here. These include shape, or contour trackers, appearance, or template trackers, and some that track both quantities, as we do.

Contour trackers maintain a representation of the shape of an object as it changes over time [40, 4, 6, 7, 32, 13, 54]. Some names given to such models include active, dynamic, and deformable contours, snakes, and active splines. These methods seek to maintain a parameterized model of the changing boundary of an object. The parametrization is typically a set of points equally spaced around the boundary of the region to be tracked. Contours are fit using local gradient information, in the vicinity of the tracked contour. Because the detection and tracking is primarily concerned with the boundary and ignore the interior of the region, such methods can get easily distracted by clutter, or visual distractors located near the boundary of the tracked object. This causes the tracker to "latch on" to another object, losing its target. Several methods have been proposed to deal with this problem, including the use of regularization terms [13, 54] and through a two-step registration procedure involving a template-matching step followed by a contour-refinement step [4].

Models of shape based on learned manifolds, such as active shape models [66], help alleviate the problem of clutter by constraining the modes of deformation the shape can take based on training data. However, these models are difficult to use in general, as they require prior knowledge about the types (and shapes) of objects to be tracked, as well as assuming that all such objects will have shape deformations that cluster.

Template-based approaches bypass the issue of shape entirely and focus instead on tracking and modeling appearance [58, 5, 29, 36, 53, 14]. This is not to say that they do not have a shape component. They must, in fact. But the shape is typically simple, such as a rectangle or ellipse, and not the focus of the representation used in tracking.

Combined models of shape and appearance are most relevant to the model suggested here. In [22], Fitzgibbon decomposed the motion of a "nowhere-static" scene into a global, rigid component, and a stochastic component, similar to those used in dynamic textures [18]. Because the global motion model was of pure translation, he was able to use a global search mechanism, enumerating out all possible translations. However, such a method does not scale well to more complex models of deformation. Active appearance models are also closely related [20, 47, 16]. AAMs model shape and appearance variation in class-specific low dimensional subspaces and are generalizations on the concepts of active shape models [66] and Eigenface models [68, 5]. Trackers based on these models (e.g. [27, 57]) suffer under the same constraints as active shape models. Namely, they require labeled training data for each class of object and can be used only to track those objects. They also require the shape and appearance of the object to vary in preset ways and, hence, would be unsuitable to track many of the objects shown in Figure 1-3, which change shape and appearance in complex, nonrepeating ways.

In contrast to the methods described above, the method we propose models shape and appearance jointly, and tracks the changes to such parameters in an online estimation framework. In this way, we are able to track regions that undergo complex, nonrepeating changes to both texture and shape. As with the models described above, we too constrain the ways in which a tracked region can change. But we do so through the use of shape priors, which effectively limit the amount of change that the model can undergo from frame to frame in an online framework that makes it suitable for general online tracking.

2.3 Feature Detection and Data Association

Section 2.1 described how to estimate and maintain the state $\hat{x}_t$ of a tracked object, given measurements z_t derived from it. In state estimation the measurements are given, presumed

to have already been taken and awaiting integration. In tracking, they are not. In vision applications, the measurements are typically derived from the visual appearance of the object within the imagery I_t at each time t. However, a significant problem in deriving these features is that we do not know where the object is. If we did, then the problem of tracking it would already be solved. The two main approaches to tracking described in chapter 1 – blob-based tracking and model-based tracking – handle this problem in two different ways.

In blob-based tracking, the silhouettes of objects are obtained via background subtraction, optical flow, or some other motion-based agglomerative process [72, 63, 67, 55]. Features, such as area, centroid position, aspect ratio, bounding box, etc., are then derived from each blob, independent of the currently tracked objects. The problem in blob-based tracking, then, is the problem of data association: which blobs and associated feature sets go with which tracks? Although there are many ways in which to assign blobs to tracks, they all typically use the predictions of state to facilitate the match. Let $T = \{\{x_0, \mathbf{P}_0\}, \{x_1, \mathbf{P}_1\}, ..., \{x_{n-1}, \mathbf{P}_{n-1}\}\}$ be the current predicted states of n tracked objects. Let $Z = \{z_0, z_1, ..., z_{m-1}\}$ be the feature sets associated with m extracted blobs. The Mahalanobis distance can then be used to measure the similarity between or, in a statistical sense, the likelihood of, any measurement z_i and track $T_j = \{x_j, \mathbf{P}_j\}$ according to

$$d(T_j, z_i) = (z_i - \mathbf{H}x_j)^T(\mathbf{H}\mathbf{P}_j\mathbf{H}^T + \mathbf{V})^{-1}(z_i - \mathbf{H}x_j). \tag{2.18}$$

The challenge in model-based tracking is different, with a solution that makes different use of the tracked states. In model-based tracking, the detection process that gives rise to measurements is not independent of the tracks. For image-based models, such as active appearance models [16], active blobs [57], template-based trackers [53, 43, 2], and the model described in this dissertation, the tracked state consists of parameters that describe the expected appearance of the model within the imagery. Thus, the challenge with model-based tracking is not in data association, which arises because the detection process is independent of the tracks, but in the detection process itself.

The process of detection is specific to the modeling approach taken. For example,

contour trackers [6, 7, 32, 13] search out along normals to the estimated curve in order to find features, such as edge features, flow vectors, or areas of high contrast, to locate the curve in the current image. Two-frame, template trackers search for instances of the tracked object in proximity to its previously detected location. Subspace methods have various ways of fitting the model to imagery, but are initialized similarly.

The approach to detection taken in this work is based on registering the model with the imagery subject to a shape prior, the details of which are given in Chapter 3 and Section 4.4. Similar to the approaches taken in contour tracking and two-frame image-based trackers, we initialize the detection process based on a prediction of shape and appearance derived from the model. The method of detection, however, is more closely related to those used by subspace methods [16, 57, 47]. Subspace methods use an iterative refinement approach to detection, subject to constraints on deformation imposed by the assumption of a learned compact shape class. The constraints we impose are based instead on shape priors, which place limits on the amount of deformation an object can undergo from frame to frame and do not require training data. In this manner, we are able to represent and track novel objects and objects whose shapes change in complex ways that do not necessarily cluster in some space of shapes.

Thus, the process of feature detection is the process of obtaining the measurements that are used within the context of recursive state estimation, in order to estimate certain properties of the tracked objects. Blob-based approaches typically arrive at such measurements independently of the tracking data, and so need to associate measurements with tracks. On the other hand, model-based approaches use the tracked states to facilitate the acquisition of measurements, and so bypass the issue of associating measurements with tracks. Though differing in their method of obtaining measurements, the measurements themselves are used similarly. They are used to update a representation of the tracked object, be it size, centroid position, or bounding box, commonly estimated in blob-based trackers, or a measure of shape, contour, or appearance, as estimated in various model-based approaches.

Chapter 3

Image Registration

In the approach to tracking described here, image registration is the process used to fit the model estimated through time $t-1$ to the new imagery at time t. In this way, we detect the region and obtain measurements of its shape and appearance, which can then be used to obtain a more refined, more accurate, model of its shape and appearance.

Fundamentally, image registration is a problem in matching. Given two images I_1 and I_2 of the same object, such as a car, we often wish to find a dense, pixelwise mapping of the object in I_1 with I_2. Take, for example, the car shown in Figure 3-1. The car was imaged at two different times and, as a consequence of it moving, appears in two different places. In the context of image registration, we would like to be able to determine, for all pixels (x, y) of the car in I_1, the corresponding pixels (X, Y) of the car in I_2.

A naive solution to this correspondence problem is to do patch matching. Suppose we wish to find the corresponding pixel (X_i, Y_i) in I_2 for known pixel (x_i, y_i) in I_1. We can take a window of pixels around (x_i, y_i), call it $w_{(x_i,y_i)}$, and find the position (X_i, Y_i) of the corresponding patch $W_{(X_i,Y_i)}$ in I_2 that best matches it:

$$(X_i, Y_i) = \underset{(X,Y)}{\operatorname{argmin}} \sum_x (W_{(X,Y)}(x) - w_{(x_i,y_i)}(x))^2. \tag{3.1}$$

Said less formally, a patch of pixels around i, perhaps a 7×7 patch, is taken from I_1. Its center is then positioned over every pixel in I_2 and a sum-of-squared-differences (SSD) is performed to determine how similar the two patches are. The pixel location over I_2 that

results in the lowest score is chosen to be the corresponding location of (x_i, y_i) in I_2.

Although this is a very simple, and intuitive, method of performing image registration, it suffers in a number of ways. For example, since every pixel is registered independently, there is no mechanism for ensuring that matches are consistent. In our car example, for instance, there is no mechanism for ensuring that the front wheel in I_1 matches the front, rather than the rear, wheel in I_2.

Furthermore, if the process of registering one image to another is thought of as a process of *deforming* one image to look like another, then the naive method described above can be thought of as a potentially very complex model of deformation, in which each pixel can go anywhere, unconstrained by any other potential match. Again, using the example of the car, this could lead to a front wheel matching a back wheel, which in the language of deformations, would represent a spectacular distortion of the car. It could also lead to two pixels on a door panel swapping positions, which makes little or no physical sense.

So, how do we overcome these limitations in the framework described above? Our solution lies in treating the registration problem as a problem in estimation, where the quantities to be estimated are the parameters of a deformation function that will bring I_1 into correspondence with I_2. When treated as an estimation problem, it is easy to see that the naive method described above leads to overfitting: there are too many parameters to be estimated (each corresponding location (X_i, Y_i)) from too little data (I_1 and I_2). In fact, there are twice as many parameters to be estimated as numbers of pixels to be matched!

Combine this with the fact that most things we wish to register do not undergo such arbitrarily complex deformations, and the above solution looks even worse. For example, the car from Figure 3-1 undergoes a simple translation. This implies that the parameters of a deformation model that can be estimated to bring I_1 into alignment with I_2 are the two parameters $(\Delta x, \Delta y)$ that indicate the car's displacement in I_2 with respect to its position in I_1. Surely it is far easier, more efficient, and will lead to more robust solutions, to estimate the two parameters of that deformation model that will bring I_1 into alignment with I_2 than if we matched each pixel independently.

So now the question becomes, "Given a particular model of deformation, how do we estimate those parameters that will bring I_1 into correspondence with I_2?" That process

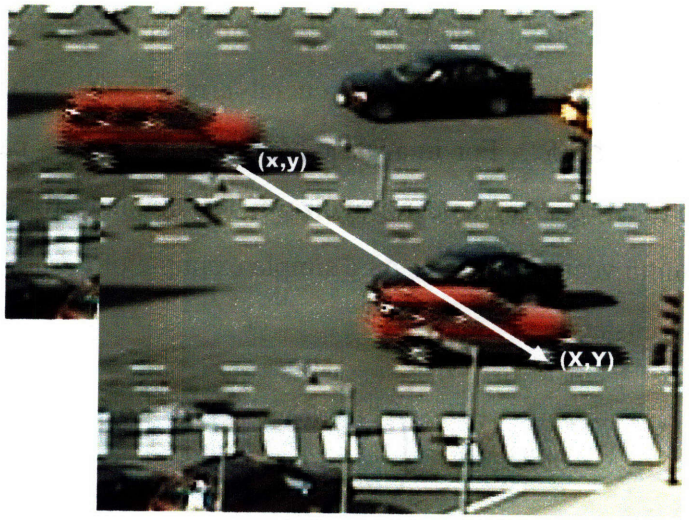


Figure 3-1: Image registration is the process of matching pixels (x, y) in one image with corresponding pixels (X, Y) in another. In this example, a correspondence is shown between a pixel belonging to the car's wheel in two images taken at different times.

was mathematically formalized in [44]. In [44], images were treated as scalar functions over the two dimensional input domain $\Omega \subset \Re^2$ and the registration process was reduced to a search over some space of deformations of Ω. Since then, that mathematical framework has been extended to incorporate more complex families of deformations, such as affine [2], piecewise affine [47], and thin-plate spline warps [8, 42]. Several variants have also been proposed to improve the overall speed of the approach, with some making simplifying [16, 57], though in general erroneous [47], assumptions, and others showing equivalence (to a first-order approximation) [1].

3.1 Registration of Scalar Images

The problem of registering scalar valued images I_1 and I_2 over the domain $\Omega \subset \Re^2$ reduces to the problem of finding the parameters p of a warp function w over Ω such that

$$g(u) = f(w(u, p)). \tag{3.2}$$

We acknowledge the functional nature of images, an assumption that will be exploited extensively, by referring to them in the more general terms f and g. Their input is defined over $u \in \Omega$, where $\Omega \subset \Re^s$. For images, typically $s = 2$. The parameters p of the input domain warp function w are a vector of size n. The output of the vector-valued warp function w is a column vector of size s. As a simple example of a warp function, consider the family of rigid translations:

$$w(u,p) = \begin{pmatrix} x + p_1 \\ y + p_2 \end{pmatrix} = u + p, \tag{3.3}$$

where $u = (x,y)^T$ and $p = (p_1, p_2)^T$. The first element of the output of w describes the warping of the x- coordinate and the second describes the warping of the y-coordinate. Under this model, we assume that g is some translated version of f. Therefore, the parameters we seek are the displacements $\hat{p} = \{\Delta x, \Delta y\}^T$ for which Equation 3.2 holds.

Rarely will it be the case, however, that there exists a $\hat{p}$ such that $g(u) = f(w(u,\hat{p}))$. Writing Equation 3.2 another way:

$$g(u) - f(w(u,p)) = 0, \tag{3.4}$$

we see that it requires of the warp estimate $\hat{p}$ that the difference between the warped f and g be zero everywhere, an unlikely occurrence. Therefore, we relax this constraint by defining the cost of a particular warp p as:

$$C(p) = \sum_u (f(w(u,p)) - g(u))^2, \tag{3.5}$$

and defining $\hat{p}$ to be the least squares solution:

$$\hat{p} = \underset{p}{\operatorname{argmin}}\, C(p). \tag{3.6}$$

In order to find, or simply approximate, $\hat{p}$, it is helpful to see how C changes in an area

around a particular choice of parameter settings p:

$$C(p+\Delta p) = \sum_u (f(w(u,p+\Delta p)) - g(u))^2. \tag{3.7}$$

Letting $F(u,p) = f(w(u,p))$ and taking a first-order Taylor expansion of F about p, we obtain

$$C(p,\Delta p) = \sum_u \left(F(u,p) + \left.\frac{\partial F}{\partial p}\right|_{(u,p)} \Delta p - g(u) \right)^2. \tag{3.8}$$

Substituting $f(w(u,p))$ back in for F and reordering yields

$$C(p,\Delta p) = \sum_u \left(f(w(u,p)) - g(u) + \left.\frac{\partial f}{\partial w}\right|_{w(u,p)} \left.\frac{\partial w}{\partial p}\right|_{(u,p)} \Delta p \right)^2, \tag{3.9}$$

where $f(w(u,p)) - g(u)$, often referred to as the error image [1, 47, 2], describes the local difference between the warped f and the target g. Using the convention that the derivative of a function with respect to a vector of variables is laid out in a row rather than a column, $\frac{\partial f}{\partial w}$ is a row vector of size s, $\frac{\partial w}{\partial p}$ is a $(s \times n)$ matrix, and Δp is a column vector of size n. Since the summation to arrive at C is over u and not w, the warped input domain, and since $\frac{\partial f}{\partial w}$ is a partial derivative w.r.t. w and is evaluated at w, $\frac{\partial f}{\partial w}$ is computed by taking the gradient of f and warping it according to $w(u,p)$. More specifically, to achieve an efficient implementation, f and $\frac{\partial f}{\partial w}$ are given, and computed, respectively, in the domain of f and then *inverse warped* onto the domain of g. The warp Jacobian, $\frac{\partial w}{\partial p}$, describes how the warp function changes as a function of its parameters, p. Continuing with the simple example warp given in Equation 3.3,

$$\frac{\partial w(u,p)}{\partial p} = \frac{\partial (u+p)}{\partial p} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \tag{3.10}$$

where the top row describes the warping of the x-coordinate w.r.t. all $p_i \in p$ and the bottom row for the y-coordinate. For rigid translation, therefore, $\frac{\partial w}{\partial p} = \mathbf{I_2}$. Cleaning up the

notation, equation 3.9 reduces to:

$$C = \sum_u \left(f(w(u,p)) - g(u) + \nabla f \frac{\partial w}{\partial p} \Delta p \right)^2. \tag{3.11}$$

When f is locally smooth, p is close to the optimal warp, and Δp is small, equation 3.11 is a good approximation to Equation 3.7. The problem of finding $\hat{p}$ is now reduced to finding the best direction Δp to move from p in order to minimize C. To find the Δp that minimizes C, we take the derivative of 3.11 with respect to Δp and set it equal to 0:

$$\frac{\partial C}{\partial \Delta p} = \sum_u 2 \left(\nabla f \frac{\partial w}{\partial p} \right)^T \left(f(w(u,p)) - g(u) + \nabla f \frac{\partial w}{\partial p} \Delta p \right) = 0. \tag{3.12}$$

Collecting terms yields the following for Δp:

$$\Delta p = \left[\sum_u \left[\nabla f \frac{\partial w}{\partial p} \right]^T \left[\nabla f \frac{\partial w}{\partial p} \right] \right]^{-1} \sum_u \left[\nabla f \frac{\partial w}{\partial p} \right]^T [g(u) - f(w(u,p))]. \tag{3.13}$$

Unless f really does depend linearly on u everywhere, Equation 3.11 only approximates 3.7 and so Δp will only give approximately the correct step. It does, however, imply an iterative approach:

$$p^{i+1} \leftarrow p^i + \Delta p^i, \tag{3.14}$$

which, when iterated over until convergence, or until $i > \eta$, some maximum number of iterations, yields an approximation to Equation 3.6 [44, 2]. This is due to the approximation of Equation 3.7 used and the derived solution of the Δp yielding a gradient-descent based method of solving for the optimal set of warp parameters.

3.2 Constraints on Terms of Cost Function

Although Equation 3.13 looks complicated, note that it only depends on a few terms: The two images, f and g, the gradient of f, ∇f, and the Jacobian of the warp function, $\frac{\partial w}{\partial p}$.

In terms of formal constraints, f, g, and ∇f can take on any real value. However, since the cost function C of Equation 3.5 is defined in terms of individual pixel differences, it is

advantageous for the images to be textured and not flat shaded everywhere. Furthermore, due to the linearization used in the cost function in Equation 3.9, it is also helpful for the images to be locally smooth. This ensures that Equation 3.9 is a good approximation to Equation 3.7.

The last term that is required to compute Δp in Equation 3.13 is the warp Jacobian $\frac{\partial w}{\partial p}$. The warp function itself defines the family of warps over which to search to align f and g. It is chosen based on some a priori expectation on the types of deformations likely to be seen. There are two constraints on the family of warps. The first is that w must be differentiable w.r.t. its parameters p everywhere in the domain Ω over which the optimization to find Δp takes place. This ensures that $\frac{\partial w}{\partial p}$ is defined everywhere. The second is that the warp function must itself be defined everywhere in Ω. This ensures that $f(w)$ is defined everywhere.

Note that these constraints arise because of the gradient descent-based method used to estimate the parameters that minimize Equation 3.5, rather than as being intrinsic to the problem of finding such an optimal set of warp parameters. For instance, enumerating out the set of all possible warp parameters in order to find the best setting would not require that the warp function be differentiable w.r.t its parameters. Likewise, an inverse-compositional approach [1] to optimizing Equation 3.5 requires[1] that the warp function form a group, a constraint not imposed by the method of optimization employed here.

3.3 Registration of Vector Images

We begin with the concepts and definitions introduced in Section 3.1. However, we now let $f(u)$ and $g(u)$ be two vector-valued functions of position vector u. To register vector-valued functions, one can generalize the cost function of equation 3.5 by using the L_2 vector norm:

$$C = \sum_u (f(w(u,p)) - g(u))^T W (f(w(u,p)) - g(u)). \tag{3.15}$$

[1] Without introducing other simplifying assumptions.

Now, f and g are vector-valued, returning column vectors of size m and the new term, W, is a symmetric weight matrix. The addition of a weight matrix allows for a linear mixing of the dimensions of the vector functions f and g. A diagonal W, for instance, could be used to give different weights, or importance, to the various dimensions of the data. A symmetric, though not necessarily diagonal, weight matrix W could represent an inverse covariance matrix, in which case the L_2 norm in Equation 3.15 becomes a Mahalanobis distance. This implies a statistical interpretation of f and g. W can also be considered a basis transform, useful if f and g are functions that represent the same phenomena but under different modalities or using different sensors.

Proceeding similarly to Section 3.1, we see how C changes in a region Δp around p and, by taking the first-order Taylor expansion of f about point p, we obtain:

$$C = \sum_u \left(f(w(u,p)) + J\frac{\partial w}{\partial p}\Delta p - g(u) \right)^T W \left(f(w(u,p)) + J\frac{\partial w}{\partial p}\Delta p - g(u) \right), \tag{3.16}$$

where J is the $(m \times 2)$ Jacobian of f, representing the derivatives of each image channel with respect to the x- and y- coordinates. For clarity, let $K = \frac{\partial w}{\partial p}$. Expanding out Equation 3.16 yields:

$$C = \sum_u \begin{pmatrix} f^T W f + f^T W J K \Delta p - f^T W g + \\ \Delta p^T K^T J^T W f + \Delta p^T K^T J^T W J K \Delta p - \Delta p^T K^T J^T W g + \\ -g^T W f - g^T W J K \Delta p + g^T W g \end{pmatrix} \tag{3.17}$$

Taking the derivative of Equation 3.17 w.r.t. Δp and setting equal to $\mathbf{0}$ yields:

$$\frac{\partial C}{\partial \Delta p} = \sum_u 2K^T J^T W f - 2K^T J^T W g + 2K^T J^T W J K \Delta p = \mathbf{0}. \tag{3.18}$$

Solving for Δp and substituting $\frac{\partial w}{\partial p}$ back in for K yields:

$$\Delta p = \left[\sum_u \left[J\frac{\partial w}{\partial p} \right]^T W \left[J\frac{\partial w}{\partial p} \right] \right]^{-1} \sum_u \left[J\frac{\partial w}{\partial p} \right]^T W \left[g(u) - f(w(u,p)) \right]. \tag{3.19}$$

This update rule is quite similar to Equation 3.13, bearing only two differences. The

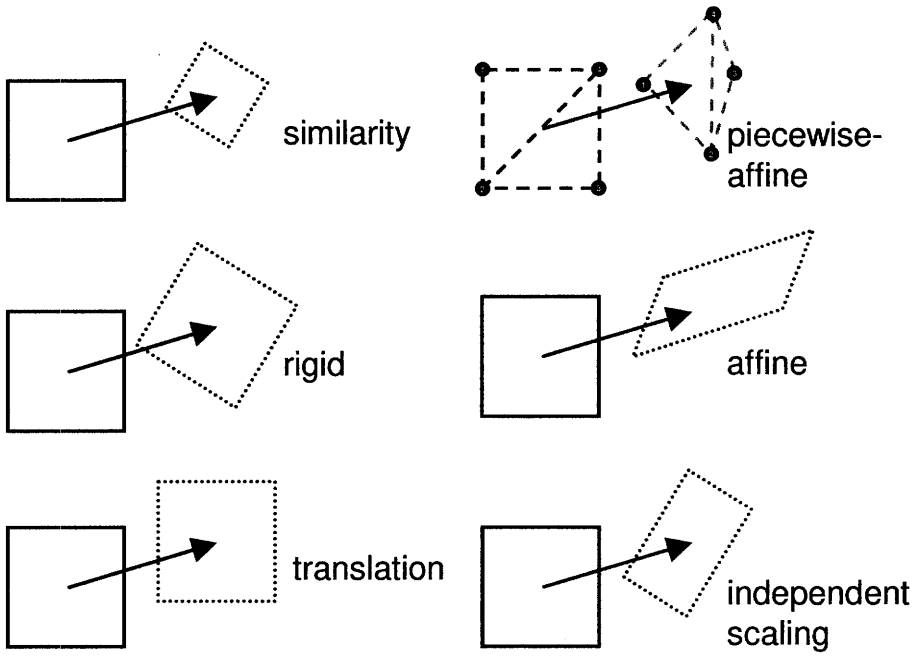


Figure 3-2: Examples of warp functions and the effects they can have on a square shape. Translation simply moves the shape. Rigid transforms add rotation. Similarity transforms add scale. A generalization off the similarity transform is to allow independent scaling of the axes. Affine transforms add skew, so the axes need not be orthogonal. Piecewise affine transforms apply affine transforms locally to achieve complex deformations.

first is the introduction of the weight matrix W and the second is that $\frac{\partial f}{\partial w}$, the gradient of f warped according to w, is now J, the Jacobian of f, laid out as a $(m \times 2)$ matrix.

3.4 Warps

In order to use the registration framework described previously, a suitable warp must be chosen that will deform the domain of f and bring it into alignment with g. The choice of warp will depend on the types of deformations that the imagery are likely to undergo. Regardless of the complexity of the deformation, however, all warps suitable for use in this framework must have two terms defined: the warp, $w(u, p)$, and its Jacobian, $\frac{\partial w}{\partial p}(u, p)$. In this section, we will describe several warp functions of increasing complexity and derive their Jacobians.

3.4.1 Translation

Aside from the identity warp, the family of rigid translations is perhaps the simplest possible. Here, we assume that f is some translated version of g. The goal then, encoded in Equation 3.6, is to find the shift in the x- and y- directions that, when applied to f, will minimize the pixelwise error between it and g. A rigid translation will warp the domain of f according to:

$$\begin{aligned} X &= x + \Delta x \\ Y &= y + \Delta y \end{aligned} . \tag{3.20}$$

The derivative of this system of two equations w.r.t. the parameters Δx and Δy is therefore

$$\begin{aligned} \frac{\partial X}{\partial \Delta x} &= 1 \quad \frac{\partial X}{\partial \Delta y} = 0 \\ \frac{\partial Y}{\partial \Delta x} &= 0 \quad \frac{\partial Y}{\partial \Delta y} = 1 \end{aligned} . \tag{3.21}$$

Under the standard notation used in this chapter, and using the language of warps and image registration, rigid translation is a warp defined by a function $(X, Y)^T = w(u, p)$ of the image domain $u = (x, y)^T \in \Omega$ and of the warp parameters $p = (\Delta x, \Delta y)^T$. In this context, the above systems of equations can be rewritten as

$$w(u, p) = \begin{pmatrix} u_1 + p_1 \\ u_2 + p_2 \end{pmatrix} = u + p, \tag{3.22}$$

and

$$\frac{\partial w}{\partial p} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \mathbf{I}_2. \tag{3.23}$$

3.4.2 Translation, Rotation, and Scaling

One way to generalize the transformation of shape described above is to allow the shape to rotate as well as translate. Such warps are called rigid transforms since they preserve angles and line lengths but not position or orientation. Figure 3-2 shows an example of such a transform. To change the size of a shape and not just its position or orientation, we can allow the shape to undergo a scale change. When the scaling of the axes are equal,

we call that a similarity transform, since the lengths of all edges are simply multiplied by that scale factor. The most general form of this class of transform is to combine rotation, translation, and independent scaling of the coordinate axes. One way to parameterize such a warp is through the composition of the three simpler warps:

$$w(u,p) = SRu + t, \tag{3.24}$$

where R is the 2×2 rotation matrix

$$R = \begin{pmatrix} \cos p_1 & -\sin p_1 \\ \sin p_1 & \cos p_1 \end{pmatrix}, \tag{3.25}$$

S is the 2×2 scaling matrix

$$S = \begin{pmatrix} p_2 & 0 \\ 0 & p_3 \end{pmatrix}, \tag{3.26}$$

and t is the translation vector

$$t = \begin{pmatrix} p_4 \\ p_5 \end{pmatrix}. \tag{3.27}$$

Taken together, the resulting equation for the warp is

$$w(u,p) = \begin{pmatrix} p_2 \cos p_1 & -p_2 \sin p_1 \\ p_3 \sin p_1 & p_3 \cos p_1 \end{pmatrix} u + \begin{pmatrix} p_4 \\ p_5 \end{pmatrix}. \tag{3.28}$$

The Jacobian is therefore

$$\frac{\partial w}{\partial p} = \begin{pmatrix} -p_2(u_1 \sin p_1 + u_2 \cos p_1) & u_1 \cos p_1 - u_2 \sin p_1 & 0 & 1 & 0 \\ p_3(u_1 \cos p_1 - u_2 \sin p_1) & 0 & u_1 \sin p_1 + u_2 \cos p_1 & 0 & 1 \end{pmatrix} \tag{3.29}$$

3.4.3 Affine

Affine transforms are a generalization of the transform introduced in Section 3.4.2 which allows for an additional *skewing* of the coordinate frame. Figure 3-2 shows an example of

an affine transform. Affine transforms preserve parallelism, but not angles.

An affine transform is one involving translation, rotation, scaling, and shearing of the input domain. For two-dimensional images, affine warps are defined by the linear relation:

$$w(u,p) = \begin{pmatrix} X \\ Y \end{pmatrix} = \begin{pmatrix} p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ 1 \end{pmatrix}, \tag{3.30}$$

where the inputs u are specified using *homogeneous* coordinates [24]. The "vector" composed of $(p_3, p_6)^T$ represents the translation parameters, as before. The 2×2 matrix composed of p_1,p_2,p_4, and p_5 hiding within the larger 2×3 matrix above represent the rest of the atomic transforms comprising the affine warp: rotation, scaling, and shearing. Parameterizing the affine warp in this way greatly simplifies its use, as the Jacobian becomes trivial to compute:

$$\frac{\partial w}{\partial p} = \begin{pmatrix} u_1 & u_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & u_1 & u_2 & 1 \end{pmatrix}. \tag{3.31}$$

Parameterized in this way, the affine warp is linear in p, and so p vanishes from the Jacobian. Compare this with the warp in Section 3.4.2, which ostensibly is a less complex warp, having fewer degrees of freedom, but, due to the nonlinear use of its parameters, results in a more complex Jacobian. Not only is this anachronistic, which by itself means nothing, but it also results in a less efficient registration process. Every iteration of Equation 3.14 updates the warp parameters p. Therefore, having $\frac{\partial w}{\partial p}$ depend on p requires that it be recomputed after every update, a costly computation. The fact that the affine transform has a Jacobian that does not depend on its parameters and, indeed, has a remarkably simple form, has not been overlooked by the vision community.

3.4.4 Piecewise Affine Warps

The family of piecewise affine deformations have been used extensively in computer vision due to their ability to model arbitrarily complex deformations and their relatively straightforward and efficient implementation. As shown in Figure 3-3, piecewise affine warps are

graphs $M = (V, E)$ whose edge structure describes a tesselated mesh. Though M is not defined in terms of physical location, we often wish to talk about it at a particular position within an image. In this regard, let $M(p)$, $p = [x_1, y_1, x_2, y_2, ..., x_v, y_v]^T$, $v = |V|$, refer to an *instance* of M with nodes at the physical location specified by p. Furthermore, let $\Omega(M(p))$, or simply $\Omega(M)$ when the context is clear, refer to the domain over which $M(p)$ is defined. Likewise, let M^i indicate a mesh element i defined as a 3-clique of nodes within M. Let $M^i(p_i), p_i = [x_1, y_1, x_2, y_2, x_3, y_3]^T$ indicate mesh element i with physical location specified in terms of its location vector p_i. Here, the indices on the coordinates are relative to element i and specify an (arbitrary) ordering of the nodes of M^i. The individual triangular elements M^i of the mesh define regions $\Omega(M^i(p_i))$ whose deformations are modeled as locally affine. As described in Section 3.4.3, an affine transform is one involving translation, rotation, scaling, and shearing of the input domain. It can be shown (see Appendix A for details) that any two nondegenerate triangles describe a unique affine transformation. Therefore, given two well-formed (i.e. containing no degenerate elements) instances $M_1 = M(q)$ and $M_2 = M(p)$ of some mesh M, the warp function $w(u, p)$ that brings points $u \in \Omega(M_1)$ to $U = w(u, p) \in \Omega(M_2)$ is:

$$w(u, p) = A_i \begin{pmatrix} u \\ 1 \end{pmatrix}, u \in \Omega(M_1^i), \tag{3.32}$$

where i is the index of the mesh element M_1^i containing u and A_i is the affine transform uniquely identified by M_1^i and M_2^i. Note that w takes points in $\Omega(M_1)$ to points in $\Omega(M_2)$, though w is only a function of p and not also q. The warp function is defined implicitly in terms of q and explicitly in terms of p. The reason for this is that, during the registration process, only the p change and the q remain fixed. The q are a part of the formulation of w, making w specific to each particular registration task.

For the purposes of *applying* this warp, as in the case above when q and p are known, Equation 3.32 implies that the individual mesh elements can be considered independently of all others. However, the coupled nature of the affine transforms in the mesh become apparent during the registration process, where the goal is to search over the space of deformations in order to align f and g.

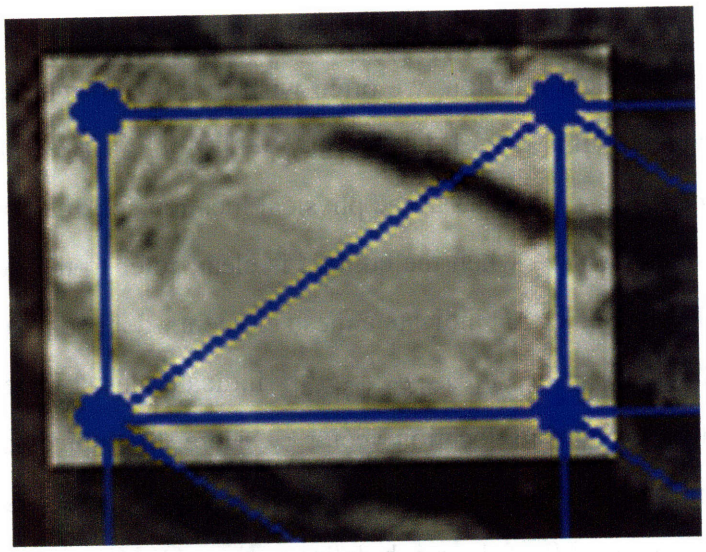


Figure 3-3: Piecewise affine warps couple neighboring affine deformations through the shared use of vertices. Though each individual mesh element is modeled as a simple affine transformation, taken together, they are capable of modeling arbitrarily complex deformation fields. In this Figure, two elements of a larger lattice are shown coupled through the shared vertices in the lower-left and upper-right corners.

3.4.5 Piecewise Affine Registration

Using piecewise affine transforms as the warp function in the registration framework described in Section 3.1 requires two quantities to be known: the warp function $w(u, p)$ and the Jacobian of the warp function $\frac{\partial w}{\partial p}$. Suppose we wish to register some mesh M instantiated at some position $q = [x_1, y_1, x_2, y_2, ..., x_v, y_v]^T$ on image I_1 with the imagery in I_2. The parameters p of $w(u, p)$ are the coordinates of the instance of $M(p)$ overlayed on I_2.

Note the role the imagery I_1 and I_2 play in the context of the functions f and g of Section 3.1: $f = I_2$ and $g = I_1$. Although w brings points $u \in \Omega(M(q))$ to $U \in \Omega(M(p))$, the cost associated with such a p in Equation 3.6 is defined over the domain $u \in \Omega(M(q))$. Thus, although w brings u to U, the effect is that the imagery of I_2 is warped back onto the coordinate frame of I_1 and all computation takes place in that domain.

The piecewise affine warp function w was described previously in Equation 3.32, Section 3.4.3, and Appendix A. It was shown that, for a given u, the warp to bring u to U depended only on the parameters of the mesh element to which u (and therefore U) belong. To be more specific, let M^i be the element of $M(q)$ that contains u. That is,

Figure 3-4: The Jacobian of the warp function w are the partial derivatives of it with respect to the warp parameters p (the target (X, Y) coordinates of the mesh). They are largest at their respective vertices, with a value of 1, and diminish linearly until they vanish at the other two vertices of each associated mesh element. Since the x- and y- components of w, w_x and w_y, are independent, $\frac{\partial w_x}{\partial p_{y_i}} = 0$ and similarly for w_y.

$u \in \Omega(M^i(q_i))$, where $q_i = [x_1, y_1, x_2, y_2, x_3, y_3]^T$ are the coordinates of the 3 nodes comprising M^i under the mesh instantiation $M(q)$. Likewise, let $U = (X, Y) \in \Omega(M^i(p_i))$, where $p_i = [X_1, Y_1, X_2, Y_2, X_3, Y_3]^T$, the same point in the same mesh element under the mesh instantiation $M(p)$. Combining Equations 3.32, A.3, and A.5, we have:

$$w(u,p) = \frac{1}{|\gamma|} \begin{pmatrix} X_1 & X_2 & X_3 \\ Y_1 & Y_2 & Y_3 \end{pmatrix} \begin{pmatrix} y_2 - y_3 & x_3 - x_2 & x_2 y_3 - x_3 y_2 \\ y_3 - y_1 & x_1 - x_3 & x_3 y_1 - x_1 y_3 \\ y_1 - y_2 & x_2 - x_1 & x_1 y_2 - x_2 y_1 \end{pmatrix} \begin{pmatrix} u \\ 1 \end{pmatrix}, \tag{3.33}$$

where $|\gamma| = -x_1 y_3 + x_2 y_3 - x_2 y_1 - x_3 y_2 + x_3 y_1 + x_1 y_2$. Equation 3.33 is simply the affine warp equation of Equation 3.30 given in terms of the coordinates of vertices of triangles rather than in the more familiar form.

From Equation 3.33, the form of the Jacobian is clear. For any given u, only 6 parameters from p are actually used, and so $\frac{\partial w}{\partial p(j)} = 0$ whenever $p(j) \notin p_i$. It is also clear that $\frac{\partial w_x}{\partial X_j} = \frac{\partial w_y}{\partial Y_j}$ for some vertex $1 \leq j \leq 3$ with coordinates (X_j, Y_j). Note that, although the ordering placed on the vertices is relative to the specific mesh element and not to the whole parameter vector p, it is straightforward to map them for the purposes of deriving the complete $\frac{\partial w}{\partial p}$.

The Jacobian turns out to be rather simple. As shown in Figure 3-4, for each vertex v,

$\frac{\partial w}{\partial v}$ is maximal, with a value of 1 at the vertex and diminishes linearly until it vanishes at the opposing vertices of each mesh element connected to it. This corresponds to intuition, as we would expect the warp to change more rapidly near the vertex that is perturbed and to have no effect on the mesh elements not connected to it.

The form of the Jacobian also explains the dependency of adjacent mesh elements during the registration process. During the registration process, the deformation of each mesh element is directly dependent on its neighbors through the shared use of vertices. This induces constraints on how much the individual elements of the mesh can deform in order to match available imagery. Since the deformation of one element will affect each neighboring element that share vertices, overfitting M^i may induce poor fits for all neighboring elements.

3.5 Shape Priors

Previously each shape, as parameterized by the warp parameters p, were treated as equally valid or important during the registration process. That is, no shape or setting of p was *a priori* more important, or correct, than any other potential setting. This implies that a set of warp parameters that would deform a 200×100 sized reference image of a car down to a single pixel in a target image was just as likely, important, or valid as a warp that maintained the size of the car. This may be true in some circumstances – for example, if nothing was known about the two images or how they were related. It is far more common, however, that we do know something about how the two images are related, even if we do not necessarily know the specific contents of the images.

For example, given that the two images were of the same scene imaged just milliseconds apart, we would be far more likely to choose the warp that preserved the size of the car as having intrinsically more value than the one which warped the car down to a single pixel. That does not mean this latter warp is invalid – just that we would consider it *a priori* less probable.

Using shape priors during image registration alters the search over warp parameters p by integrating the idea that some shapes, or warp parameters, are intrinsically more likely,

or more valid, than others. Conceptually, we want to promote "good" shapes and penalize "bad" ones. One way to incorporate such shape priors into the registration process is through the introduction of a set of terms to the cost function as described in Equation 3.15:

$$C(p) = \sum_u \left(f(w(u,p)) - g(u)\right)^T W \left(f(w(u,p)) - g(u)\right) + \sum_{s \in S} d_s(p)^2. \tag{3.34}$$

The new term, $\sum_{s \in S} d_s(p)^2$, depends only on the warp, or shape, parameters p, and not on the imagery f and g. It is a summation over the family S of shape priors, each with associated cost d_s. Thus, for a particular p', there will be a cost associated with the error $f(w(u,p')) - g(u)$ in matching the imagery, as well as the costs $d_s(p')^2$ associated with the choice of warp parameters p'. The more likely the shape, the smaller the cost associated with it. Thus, for two warps p_1 and p_2 which match f and g equally well, the one whose shape is more likely will incur the lesser cost, and so is the better warp.

We now focus on finding the best p in the presence of shape priors. Since the shape prior is additive, we let the rest of the derivation of Section 3.3 stand and focus on the changes caused by the addition of the shape priors. That part of the cost function C_S for which the family of shape priors are responsible changes in a local area about p according to:

$$C_S = \sum_{s \in S} d_s(p + \Delta p)^2. \tag{3.35}$$

Linearizing the d_s about p yields the approximation:

$$C_S = \sum_{s \in S} \left(d_s(p) + \frac{\partial d_s}{\partial p} \Delta p \right)^2, \tag{3.36}$$

the derivative of which, with respect to Δp, is:

$$\frac{\partial C_S}{\partial \Delta p} = \sum_{s \in S} 2 \left(\frac{\partial d_s}{\partial p} \right)^T \left(d_s(p) + \frac{\partial d_s}{\partial p} \Delta p \right). \tag{3.37}$$

Combining Equations 3.37 and 3.18 and setting the total quantity equal to 0 gives:

$$\frac{\partial C}{\partial \Delta p} = 0 = \begin{array}{l} \sum_x 2K^T J^T W f - 2K^T J^T W g + 2K^T J^T W J K \Delta p + \\ \sum_s 2\left(\frac{\partial d_s}{\partial p}\right)^T \left(d_s(p) + \frac{\partial d_s}{\partial p}\Delta p\right). \end{array} \tag{3.38}$$

Finally, solving for Δp:

$$\Delta p = \left[\begin{array}{l} \sum_x \left[J\frac{\partial w}{\partial p}\right]^T W \left[J\frac{\partial w}{\partial p}\right] + \\ \sum_s \left(\frac{\partial d_s}{\partial p}\right)^T \frac{\partial d_s}{\partial p} \end{array} \right]^{-1} \left[\begin{array}{l} \sum_u \left[J\frac{\partial w}{\partial p}\right]^T W \left[g(u) - f(w(u,p))\right] + \\ \sum_s d_s(p) \left(\frac{\partial d_s}{\partial p}\right)^T \end{array} \right] \tag{3.39}$$

Recall from Section 3.1 that the derivative with respect to a column vector is laid out as a row vector. So, $\frac{\partial d_s}{\partial p}$ is a row vector of length n, where n is the number of parameters p.

Incorporating shape priors into the cost function C induces additive shape terms, of a similar form as the data fidelity terms, into both the "numerator" and "denominator" of Equation 3.39. Since the only shape terms that appear in Equation 3.39 are d_s and $\frac{\partial d_s}{\partial p}$, the only constraint on the family S of shape priors that can be used within this framework is that each d_s must be differentiable with respect to the parameters p.

For example, suppose the warp was a piecewise affine mesh, as described in Section 3.4.4. Therefore, the parameters p are the (x, y) coordinates of the vertices of such a mesh. In the formulation of the tracker presented in Chapter 4 and results presented in chapter 5, we use a spring model prior on the piecewise affine deformation. A spring model prior applies an energy, or cost, associated with a deformation in length to each edge. Let $S = E$, so that the family of shape priors is defined over the edge set of the mesh. Each $s \in S$ is defined in terms of a single edge consisting of two nodes (q, r). Each can therefore be described by

$$d_{(q,r)\in E} = \lambda \left(\sqrt{(q_x - r_x)^2 + (q_y - r_y)^2} - s_{(q,r)} \right), \tag{3.40}$$

where $s_{(q,r)}$ is the "preferred" length of the edge and λ is a mixing coefficient used to balance the effect of the shape prior with the data fidelity term. Equation 3.34 will square the

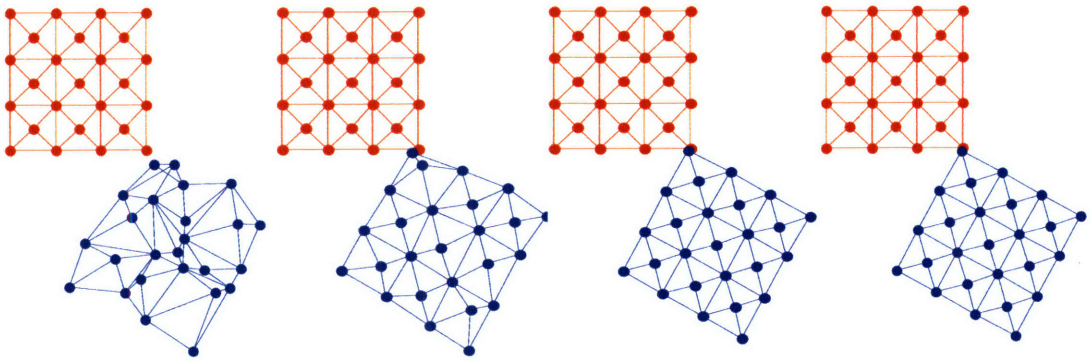


Figure 3-5: Optimizing the cost function C of Equation 3.34 in the absence of imagery illustrates the effect of the shape prior. A spring model shape prior imposed on a piecewise affine mesh regularizes the shape of the mesh. From an initially deformed condition, the optimization seeks to preserve the shape of the mesh, as indicated by the reference red mesh, though not its orientation or absolute position.

cost associated with this difference, so shortened and lengthened edges both get penalized. The Jacobians $\frac{d_{(q,r)}}{d_p}$ of each shape prior are straightforward to compute. Each has 4 nonzero elements corresponding to the parameters in p associated with q_x,q_y,r_x,r_y.

The effect of such a prior can be seen more clearly in the absence of any imagery. Figure 3-5 shows the path taken to minimize C from an initially deformed mesh on an all-white image. The input is an axis-oriented square mesh. The initial p is of a displaced, rotated, and deformed version of that mesh. The result after optimization is that the shape is preserved, though the initial displacement and rotation are not optimized or compensated for. This is because the shape prior is invariant to these conditions, optimizing only over edge length, not orientation or position. Thus, the effect of a spring model shape prior is to preserve shape, but not necessarily position or orientation.

Shape priors encode the simple idea that certain shapes are more likely to occur than others. The car is unlikely to be only one pixel in size and very likely to have its size remain unchanged. Incorporating such priors into the registration process enables the search over warp parameters to take such biases into consideration. This is done by encoding a cost associated with each set of candidate warp parameters, which influences the search over parameters to favor those that not only well align the imagery, but that do so with shapes, or deformations, that are *a priori* more probable.

Chapter 4

Tracking Dynamic Regions of Texture and Shape

In this chapter, we introduce an appearance model capable of representing dynamic, visual phenomena that undergo changes to both shape and textured appearance. We describe how such a model can be learned over time within a recursive estimation framework based on the Kalman filter. We also describe how to acquire measurements of the tracked regions by fitting the predicted shape and appearance of each region to the available imagery, taking into consideration that tracked regions may occlude one another.

It is not difficult to find examples of the visual phenomena so described. For instance, the two-dimensional projection of a car making a turn undergoes just such a change to its shape and appearance. The pose change induced by the turn causes different parts of the car to come into view, and causes other parts to be occluded from view. And, because the car is not spherically symmetric with respect to its shape or texture (i.e. door panels do not look like the hood or trunk of the vehicle), both of these properties will appear to change when imaged by a camera. To be able to model the appearance of a car as it undergoes such a routine manoeuver requires a model capable of adapting to such changes in shape and texture.

Though such a model can be used to track the visual appearance of rigid bodies such as cars, it is ideally suited to enable the tracking of amorphous, nonrigid phenomena such as groups of people. When individuals form a group, much of their motion, their trajectory,

can be expressed in terms of the motion of the group as a whole. Furthermore, the appearance of the group can be compactly represented as a dynamic, textured surface rather than as a complex arrangement of continuously occluding independent appearances. Motion of the individuals unaccounted for by that of the group as a whole, such as when two people swap places within the group, induce a change in the shape and textured appearance of the region. A model that tracks changes to both shape and appearance is able to well represent such phenomena.

The tracking of such regions in a recursive estimation framework requires several quantities to be known. The process of tracking requires:

State Space Maintaining a representation of the tracked region involves estimating the specific shape and texture parameters of that region within the space of all possible combined appearance coefficients. It is an estimate of the state of the tracked region, and the space of all possible estimates is the state space.

State Dynamic The dynamic describes how the state of the tracked region – its appearance and shape – evolves over time. It is used to predict a distribution over where in the imagery the tracked region will be located, and what its appearance will be.

Measurements The measurements are the specific instances of the regions within the imagery – the image of the car, or of the tracked crowd. They are integrated into the estimate of shape and texture of each tracked region and in this way, the model can adapt to changes in the visual appearance over time.

The mathematical framework of state estimation assumes that the measurements are given. However, for model-based visual trackers such as the one described in this chapter, the process of acquiring those measurements, through the detection of the regions within the imagery, forms an integral part of the tracking system. Figure 4-1 shows how the acquisition of measurements fits into the estimation and tracking framework. The process of tracking can therefore be described in terms of the following. At each time t:

1. Predict the appearance, shape, and location of each tracked region from the models

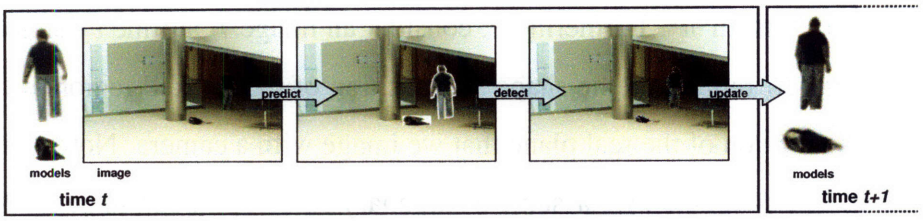


Figure 4-1: Overview of the model-based visual tracking process. Visual tracking integrates the process of detection into the predictor-corrector paradigm of state estimation [70].

maintained through time $t-1$, taking into consideration estimated occlusion boundaries.

2. Detect instances of the regions by fitting the predicted shape and appearance to the available imagery. These become the measurements at time t.

3. Update the models by integrating the new data into the state estimates and reestimating the occlusion boundaries of overlapping models.

The state space and dynamic, from which the processes of prediction, detection, and update are derived, are described in Section 4.1. Section 4.3 describes how to forward predict the model in time to obtain the a-priori estimate of state. The predicted appearance of each region is used for locating and detecting instances of the tracked regions within the imagery. Section 4.4 describes how to perform the search for instances of each tracked region using a gradient-descent based image registration method. With measurements acquired, the process of updating the tracked models is described in Section 4.5. Occlusion reasoning and its reduction to the problem of image segmentation is described in Section 4.6 along with various methods of determining occlusion boundaries derived thereof.

4.1 Model Description

Consider a contiguous region $R_t \subset \Re^2$ at time t. This could represent the visual "footprint", or support, of a car, or a crowd of people for instance. Although ultimately it will not be represented as such, it is helpful to think of R_t as a discrete set of points defined over the

two-dimensional domain $\Re^2$. Furthermore, consider an image I_t at time t defined over the domain $\Omega \subset \Re^2$. R_t represents the shape and position of the tracked region at time t and Ω represents the portion of the real plane that we image with a camera. Note that Ω has no time subscript. For simplicity, we assume a stationary camera, though in general this need not be true.

Note also that it is unnecessary for $R_t \subset \Omega$, in which case part or all of the region may lie outside the domain of the available imagery. This implies that part or all of R_t may not currently be visible. For example, the car may be leaving the scene. The part that is currently visible, however, gives us a sample of the region's appearance.

Furthermore, let α_t be the vector-valued analysis of I_t filtered through ϕ such that $\alpha_t = \phi(I_t)$, defined over the domain Ω and range $\Re^d$. ϕ represents an arbitrary filterbank that will filter each I_t, the output of which will be a vector of size d at each location $u \in \Omega$. For example, the ϕ may be $x-$ and $y-$ derivative filters, in which case $d = 2$. The ϕ could also represent the output of an edge detector. It may also represent a Gaussian or Laplacian pyramidal decomposition of the image. Or, it may simply be the identity operator, in which case $\alpha_t = I_t$. In this case, for grayscale images, $d = 1$ and for rgb images, $d = 3$.

We seek to maintain a joint model of the shape and appearance of R as it changes over time. One approach is to model the changes in appearance as a linear dynamic on the analysis coefficients alone:

$$\alpha_t(R) = \mathbf{A}\alpha_{t-1}(R) + \omega, \tag{4.1}$$

while keeping the shape, as represented by R, constant. This is analogous to the formulation of appearance used in dynamic textures [18, 59, 23]. However, for the types of visual phenomena we seek to track, the region R not only changes appearance over time, it changes shape as well. The car is turning, or the individuals within the group of people are moving apart, changing the shape of the crowd. The coordinate frame over which R is defined undergoes a deformation w such that

$$R_t = w(R_{t-1}, p). \tag{4.2}$$

Recall from Chapter 3 that p are the n parameters of the warp function w. We model

this deformation as a piecewise affine warp, parameterized by the (x, y) location of the $\frac{n}{2}$ vertices of associated mesh M, whose specific instance $M(p)$ at time t defines the region R_t. See Section 3.4.4 for a detailed description of this family of warps.

Since the region is changing shape, the parameters p of the piecewise affine warp w are also changing, though not arbitrarily. These changes can be tracked, just as the appearance coefficients were in Equation 4.1, according to the dynamic:

$$p_t = \mathbf{B}p_{t-1} + \eta, \tag{4.3}$$

where the p become time-dependent p_t and are correlated with their setting at an earlier time $t-1$ by the linear dynamic $\mathbf{B}$ and additive noise term η. Recall that the parameters p_t of a piecewise affine warp are the $x-$ and $y-$ locations of each vertex of the associated mesh. To this end, the appearance under R_t is related to its appearance under R_{t-1} by:

$$\alpha_t(R_t) = \alpha_{t-1}(w(R_{t-1}, p_t)) + v. \tag{4.4}$$

We have now defined two ways in which the appearance under R_t can be derived from the appearance under R_{t-1}. Equation 4.1 holds the shape of the region constant and changes its appearance, whereas Equation 4.4 holds the appearance coefficients constant and changes the region's shape. The appearance of each pixel under R_t can be explained as either a pixel moving from a neighboring location at time $t-1$ to its current location at time t – a deformation – or as a consequence of it not moving, but changing appearance. For example, a single pixel's change in color from $t-1$ to t may be due to the car turning, and so that pixel is the image of a different part of the car than it was at the previous time. It could also be that the pixel is the image of the same part of the car, but under changing illumination.

Indeed, the fact that the change in appearance can be the consequence of two divergent processes is a description of the latent ambiguity between shape and appearance described by Doretto in [17]. However, where [17] resolves the ambiguity through the use of a modeling responsibility term estimated during a supervised learning phase, we resolve it

by combining the dynamics into a form suitable for online estimation and tracking:

$$\alpha_t(R_t) = \mathbf{C}\alpha_{t-1}(w(R_{t-1}, p_t)) + q. \tag{4.5}$$

The shape of the region is described in terms of the parameters p of the piecewise affine warp used to model the deformations that each shape can undergo. Recall from Section 3.4.4 that the p consist of the x- and y- locations of each vertex of the mesh at each time t. The change in shape is tracked using a linear dynamic with Gaussian uncertainty according to Equation 4.3. Because the p are not location-independent, it is possible to not only track the deformation in shape of the each region, but also such statistics as position and velocity. For the results presented in Chapter 5, a constant velocity assumption is made, with the shape state estimated during the process of tracking consisting of the positions p and velocities $\dot{p}$ of the parameters of the warp. Such a dynamic can be represented as:

$$\begin{aligned} p_{t+1} &= p_t + \dot{p}_t \Delta t + \omega_1 \\ \dot{p}_{t+1} &= \dot{p}_t + \omega_2 \end{aligned} \tag{4.6}$$

The appearance of each region is similarly described in terms of the analysis coefficients α. Though in the case of the appearance coefficients, a random-walk dynamic is used, such that

$$\alpha_{t+1} = \alpha_t + \omega_3 \tag{4.7}$$

An issue that may arise in the implementation of such a model is that, as the tracked shape of the region changes so, too, do the number of tracked appearance coefficients. One way of handling this is to assume a shape-free context [16, 47] in which the appearance coefficients will be estimated. A shape-free context is one in which changes to shape and position are compensated for when considering appearance. One way to separate shape from appearance is to *choose* a reference shape and hold that constant when estimating appearance.

Similar models, such as active appearance models [16, 27, 47, 20] and active shape models [66], choose the mean shape from the set of training shapes to use as a reference shape. Two considerations preclude the use of a mean shape in the model described here.

One is that we do not have training data from which to estimate a mean shape. The other is that the shapes we track do not necessarily cluster in such a way that gives semantic meaning to the mean shape. There is no a-priori reason to suspect one shape is any better, or more representative, of a particular region than any other. Another, practical, consideration is that the only instance of a shape we have from the start of a track is its initial shape. For these reasons, the initial shape p_0 of the tracked region is used as the basis for a shape-free context $p_\alpha = p_0$.

Within a shape-free context, we can describe the shape and appearance dynamic of Equation 4.5 with a joint model such that:

$$p_t = \mathbf{B}p_{t-1} + \eta \tag{4.8}$$

$$\alpha_t = \mathbf{C}\alpha_{t-1} + v, \tag{4.9}$$

where $\eta \sim \mathcal{N}(0, \mathbf{E})$ and $v \sim \mathcal{N}(0, \mathbf{V})$. Thus, an appearance model is the four-tuple $m = \{\hat{p}, \mathbf{P}_p, \hat{\alpha}, \mathbf{P}_\alpha\}$ consisting of estimates of shape and appearance as well as the uncertainties in those estimates.

4.1.1 Reparametrization of the shape-free context

Though the shape of the tracked region at $t = 0$, p_0, is suggested as the basis for the shape-free context p_α under which appearance will be estimated, that need not be the case indefinitely. A reparametrization of the basis shape can be performed at any time. Let $\hat{\alpha}_t$ be the estimated shape-free appearance at time t and let $\mathbf{P}_t$ be the covariance of that estimate such that $E[\alpha_t] = \hat{\alpha}_t$ and $E[(\alpha_t - \hat{\alpha}_t)(\alpha_t - \hat{\alpha}_t)^T] = \mathbf{P}_t$. Furthermore, let $w(\cdot, p)$ be a warp such that $\alpha'_t = \alpha_t(w(\cdot, p))$. The effect of w is to change the reference shape under which the α will henceforth be estimated. For the purposes of application and for any warp parameterized as a deformation of the input space, it is possible to construct a matrix $\mathbf{W}$ such that $\alpha'_t = \mathbf{W}\alpha_t$. Thus, the application of the warp is a linear operation. The tracked

state can therefore be reparametrized by:

$$\hat{\alpha}'_t = \mathbf{W}\hat{\alpha}_t \tag{4.10}$$

$$\mathbf{P}'_t = \mathbf{W}\mathbf{P}_t\mathbf{W}^T. \tag{4.11}$$

With the state space in which the appearance coefficients are estimated suitably warped, the dynamic of Equations 4.8 and 4.9 can continue to be used to estimate the shape and textured appearance of a tracked region.

4.2 Measurement Model

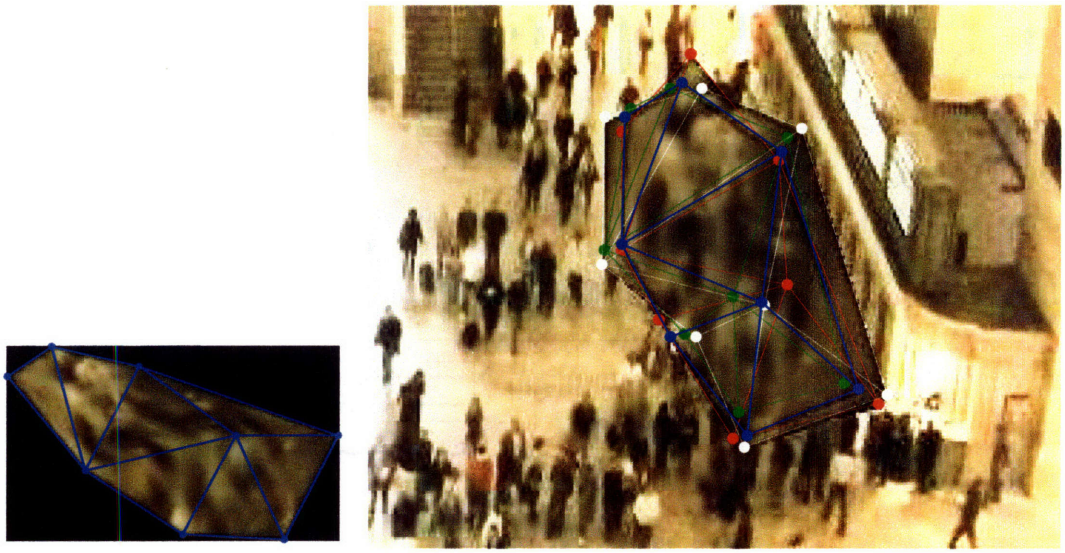
Measurements, or observations, of each tracked region m_t come from the imagery $a = \phi(I_t)$ at time t and the parameters p' of the associated piecewise affine mesh. Similar to the measurement process described by Jepson et al., in [36, 35], the measurements we use are based on the appearance coefficients a.

Let R indicate the region under mesh instance $M(p')$ such that $R = \Omega(M(p'))$. Furthermore, let $w(R, p')$ be the domain warp function that brings $a(R)$ into the shape-free context R_α in which the α are estimated, such that $R_\alpha = w(R, p')$. A measurement, therefore, consists of a shape p' and its shape-free appearance $a(w)$. The measurement model can therefore be described by:

$$p' = \mathbf{H}_p p + r \tag{4.12}$$

$$a = \mathbf{H}_a \alpha + s, \tag{4.13}$$

where the additive noise terms r and s are assumed normally distributed such that $r \sim \mathcal{N}(0, \mathbf{R})$ and $s \sim \mathcal{N}(0, \mathbf{S})$. The shape state consists of positions and velocities of the vertices of the piecewise affine mesh M associated with model m. Since the shape measurements p' consist only of positions, $\mathbf{H}_p$ is a truncated identity matrix. The appearance state consists of estimated shape-free appearance coefficients of the same form as a. Thus



(a) Region model (b) Predicted appearance of region at time t

Figure 4-2: Detecting the region within the imagery at time t begins with predicting its textured appearance and shape from the tracked model. (a) Model of shape and appearance of region estimated from imagery until time $t-1$. (b) Predicted appearance of region at time t. The blue mesh indicates the mean predicted shape, which is derived via the dynamic in Equation 4.14. The other shapes represent samples from the distribution about the mean shape, indicating the uncertainty in the prediction. Contrast of the imagery has been enhanced to aid visualization.

$\mathbf{H}_a = \mathbf{I}$.

4.3 Prediction

The estimation process progresses by forward predicting the state of each tracked model m. Predicting the shape and appearance of the tracked region involves evolving the state according to the dynamics given by Equations 4.8 and 4.9 in order to arrive at the a-priori estimate m^-:

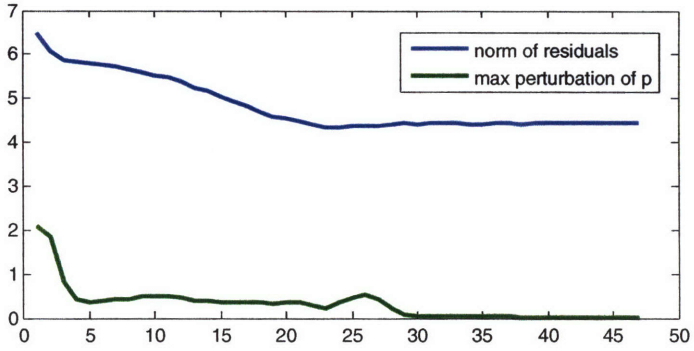
$$p_t^- = \mathbf{B}p_{t-1} \tag{4.14}$$

$$\mathbf{P}_p^- = \mathbf{B}\mathbf{P}_p\mathbf{B}^T + \mathbf{E} \tag{4.15}$$

$$\alpha_t^- = \mathbf{C}\alpha_{t-1} \tag{4.16}$$

$$\mathbf{P}_\alpha^- = \mathbf{C}\mathbf{P}_\alpha\mathbf{C}^T + \mathbf{V}. \tag{4.17}$$

(a) residuals after iteration 1,10,20,30,40,47



(b) Plot of residuals and perturbations as a function of iteration

Figure 4-3: Fitting model to imagery involves iteratively solving for parameters p of the warp that brings them into alignment, according to Equations 4.20 and 4.21. (a) visualization of $\mathbf{E}$, the error between model and imagery at different stages in the registration process. (b) plot of L_2-norm, or the sum-squared-residuals, after every iteration. Also shown is a graph of the max perturbation to p as a function of iteration. Measured in pixels, it is the largest value of the vector Δp at each iteration step. In order to accentuate the effects of the registration process, the initial position of the mesh was displaced from its predicted location and, additionally, each node's position was randomly perturbed.

Figure 4-2 shows an example of forward predicting the state of the dynamical system, including a visualization of the uncertainty in the prediction. In practice, this is performed by predicting the position and shape of the region by evolving the shape according to Equation 4.14. The appearance is evolved within the shape-free context and then warped, according to $w(\cdot, p_t^-)$, onto the predicted shape.

4.4 Detection

The process of tracking progresses via the association of measurements with tracks. The measurements we seek are instances of the tracked regions within the imagery $a = \phi(I_t)$:

the image of the car, or of the group of people. To obtain those measurements, we must first detect the objects or regions within a. With an estimate m_t^- of the location, shape, and appearance of a tracked model at time t, computed according to Equations 4.14-4.17, we have a reasonable starting point from which to locate such instances. Detection is now a process of refining that initial estimate in order to obtain a good fit of model to imagery.

Finding a good fit involves finding the n warp parameters p' that make each model best match the imagery. In the context of image registration, we wish to minimize a cost associated with a particular choice of parameters p:

$$C(p) = \sum_{u \in R_\alpha} \mathcal{E}(u)^T \mathbf{W} \mathcal{E}(u) + \lambda \sum_{f \in F} d_f(p)^2, \tag{4.18}$$

where $\mathcal{E} = \mathbf{H}_a \hat{\alpha} - a(w(R_p, p))$, the visual difference between tracked region $\hat{\alpha}$ and proposed detection $a(w(R_p, p))$. The first term is a summation over all appearance coefficients within the shape-free context R_α. With a minor abuse of notation, $\hat{\alpha}(u)$ is a vector of size d representing the predicted filter responses at location u within the shape-free context. $a(\cdot)$ is similarly sized and represents the filtered imagery warped onto R_α. The matrix $\mathbf{W}$ is as described in Section 3.3, and allows a linear mixing of the various dimensions of the data. The second term is a summation over a family of shape priors used to regularize the match and provide robustness with respect to occlusions, unavailable data (e.g. part of the mesh domain R is outside of the domain of available imagery), and visual distractors within the imagery. We use as shape priors a family of spring models which provide rigidity to the mesh. See section 3.5 for a detailed description of incorporating such priors into the registration process. The shape used is the previously estimated region shape, p_{t-1}.

The parameters we seek, therefore, are the parameters p^+ that minimize Equation 4.18:

$$p^+ = \underset{p}{\operatorname{argmin}}\, C(p). \tag{4.19}$$

By taking a first-order Taylor expansion about the warp parameters p, we approximate a solution to Equation 4.19 by iterating (until convergence) over

$$p \leftarrow p + \Delta p, \tag{4.20}$$

where

$$\Delta p = \begin{bmatrix} \sum_u [\mathbf{GJ}]^T \mathbf{W} [\mathbf{GJ}] + \\ \lambda \sum_{\{f \in F\}} \left(\frac{\partial d_f}{\partial p}\right)^T \frac{\partial d_f}{\partial p} \end{bmatrix}^{-1} \begin{bmatrix} \sum_u [\mathbf{GJ}]^T \mathbf{W} \mathcal{E} + \\ \lambda \sum_{\{f \in F\}} d_f \left(\frac{\partial d_f}{\partial p}\right)^T \end{bmatrix}. \quad (4.21)$$

The $(d \times 2)$ matrix G and the $(2 \times n)$ matrix J are the Jacobians of a and the geometric warp function w, respectively. G can be considered the generalization of an image gradient to vector-valued functions. The only constraints on the form of the warp function and the shape priors is that they must be differentiable with respect to the warp parameters p. See Chapter 3 for a derivation of Equation 4.21 and a more detailed description of image registration methods.

Figure 4-3 shows an example of the detection process. For illustrative purposes, the predicted shape was randomly perturbed, so that the initial shape used in the search would be significantly different from the optimal shape found several iterations later. As can be seen, the shape changes most rapidly in the first few iterations and quickly settles into a local minima. Once the process converges, there remain significant differences between predicted and measured appearance. These differences are due to the motion of individuals within the group of people. These motions are treated as stochastic appearance variation and are modeled as changes to the region's shape-free appearance.

When Equation 4.20 converges, we obtain an approximation p' of the best-fit parameters p^+. The detection and, hence, the measurement consists of the p', its support $R_{p'}$, and its shape-free appearance $a(w(R_{p'}, p'))$.

4.5 Update

Given a measurement consisting of positions p' of mesh vertices and shape-free appearance $a(w(R_{p'}, p')))$, we now address the problem of updating the appearance model m at time t. Updating m involves integrating the new measurement of shape and appearance into the tracked estimate of state. This can be described by:

Figure 4-4: Estimate $\hat{\alpha}$ of shape-free appearance over the course of a tracked sequence of a group of people. Time progresses in column-major order from upper-left to lower-right.

$$\hat{p} = \hat{p}^- + \mathbf{K}_p(p' - \mathbf{H}_p\hat{p}^-) \tag{4.22}$$

$$\mathbf{P}_p = \mathbf{P}_p^- - \mathbf{K}_p\mathbf{H}_p\mathbf{P}_p^- \tag{4.23}$$

$$\hat{a} = \hat{\alpha}^- + \mathbf{K}_\alpha(a - \mathbf{H}_a\hat{\alpha}^-) \tag{4.24}$$

$$\mathbf{P}_\alpha = \mathbf{P}_\alpha^- - \mathbf{K}_\alpha\mathbf{H}_a\mathbf{P}_\alpha^-, \tag{4.25}$$

with the gains $\mathbf{K}_p = \mathbf{P}_p^-\mathbf{H}_p^T(\mathbf{H}_p\mathbf{P}_p^-\mathbf{H}_p^T + \mathbf{R})^{-1}$ and $\mathbf{K}_\alpha = \mathbf{P}_\alpha^-\mathbf{H}_a^T(\mathbf{H}_a\mathbf{P}_\alpha^-\mathbf{H}_a^T + \mathbf{S})^{-1}$. All terms in the above equations are, in general, time-dependent, though the index for current time t was omitted to simplify the notation.

The above can be explained in the usual manner. The positions p' give a noisy measurement of the true position of the region at time t. This noisy measurement is integrated into our estimate of position, *and velocity*, $\hat{p}$, through the innovation $(p' - \mathbf{H}_p\hat{p}^-)$ which is scaled by $\mathbf{K}_p$. As indicated by the terms that comprise $\mathbf{K}_p$, the weight we give to the innovation is related to the uncertainties in measurement, dynamic, and state estimate. The appearance update can be explained analogously. When the measurement and process noise covariances do not change with time, the uncertainty in the estimate and the gain terms $\mathbf{K}_p$ and $\mathbf{K}_\alpha$ will quickly converge and can be computed offline [70]. This is done for the results

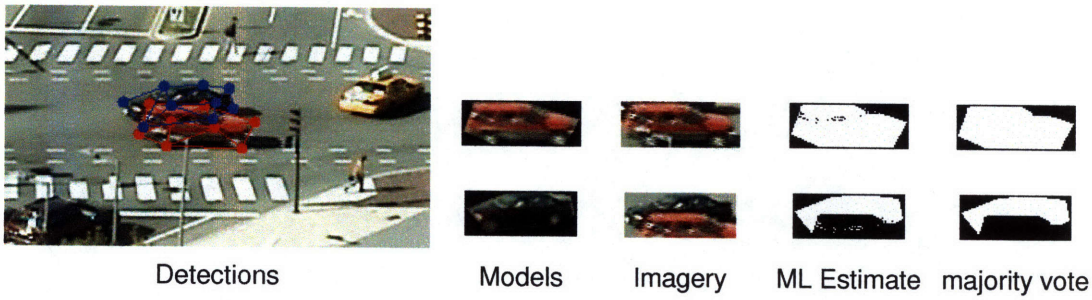


Figure 4-5: Occlusion reasoning is an integral part of the tracker described in this chapter. Given detections within the imagery and models of appearance, occlusion reasoning becomes a segmentation, or M-way classification problem. Two methods of occlusion reasoning are suggested. One is based on the maximum likelihood (ML) estimate of model responsibility given the detections. The other is a majority-vote scheme, in which the appearance under contiguous regions of occlusion is assumed to originate from only one tracked object at a time.

presented in Chapter 5.

Figure 4-4 shows an example of the evolution of the appearance estimate $\hat{\alpha}$ over the course of a tracked sequence. As the sequence progresses and the shape continues to evolve, so, too, does the appearance of the group within the shape-free context. The individuals within the group begin to lose their distinction within the shape-free appearance. By the end, almost no discernable shape is present, replaced instead by a textured appearance that represents the group as a whole.

4.6 Occlusion Reasoning

Previously, we have assumed that each tracked region m_i was always visible, such that $R_{m_i} \subseteq \Omega$ and that the m_i were never occluded. This is a requirement common to many appearance models (e.g. [17, 16, 57, 23]), though it can limit the practical application of such techniques. Layered tracking methods, such as those described by Jojic and Frey in [38] and by Winn and Blake in [71], handle occlusions explicitly, by learning the depth- or z- order of layers in an expectation-maximization framework. Unfortunately, due to the size of the search space, these types of approaches have a number of limitations. They model only simple deformations of the layers, such as translations, scalings and, more

recently, affine transforms, and their learning method is a batch process over all images in the sequence, making it difficult to apply in an online, tracking scenario. Furthermore, the z-ordering of the layers is fixed, and so it is impossible to model objects whose relative ordering changes throughout the tracked sequence.

The approach to occlusion reasoning suggested here does not maintain a full z-ordering of the m_i. Rather, for tracked regions that intersect, we estimate, for each location in the imagery, which model is currently visible and, hence, responsible for the appearance at that location. We suggest two methods for estimating such visibility masks: a per-pixel approach and a winner-take-all approach. Let $m_{1,t}, ...m_{M,t}$ be the M tracked region models at time t and let $R_{1,t}, ...R_{M,t}$ be the M detected regions. To simplify the notation and whenever the context is clear, we shall omit the time index t from such descriptions. Let $a(R_1), ...a(R_M)$ be the appearance of each detected region. Furthermore, let m_0 be the null region, with domain Ω and uniform appearance model. Let its dynamic in shape and appearance be stationary with perfect certainty.

One way to model occlusion masks is on a per-pixel basis. For each location $u \in \Omega$, we can estimate the visibility masks according to:

$$\Psi(u) = \begin{cases} \text{argmin}_{i \in [0,M]} \mathcal{L}_i(u) & \text{if } u \in \Omega \\ 0 & \text{otherwise} \end{cases}, \tag{4.26}$$

where

$$\mathcal{L}_i(u) = (a(w(u, p_i)) - \hat{\alpha}(u_\alpha))^T (\mathbf{P}_\alpha + \mathbf{S})^{-1} (a(w(u, p_i)) - \hat{\alpha}(u_\alpha)). \tag{4.27}$$

In other words, when evaluating the likelihood of $a(u)$ under model m_i, warp the imagery under R_i, according to $w(u, p_i)$, onto the shape-free context R_α for m_i and compute the likelihood of the corresponding location within the shape-free context. In this way, occlusion reasoning reduces to finding the maximum likelihood estimate for model given data. The null region, which represents the unknown, untracked regions within the imagery, has a constant likelihood. This induces a threshold on the tracked appearance models, so that poor matches of imagery to models $m_1, ...m_M$ are instead described by the null region and

considered outliers.

Another method of estimating which regions are visible is to assume that regions cannot partially occlude each other. Where two or more regions intersect, that area must be fully explained by one and only one model. Let $\bigcap = \{\cap_1, ...\cap_N\}$ be a list of domains representing the unique intersections of models $m_0, ...m_M$. For example, $\cap_1$ may represent the intersection of R_1,R_3,and R_8, while $\cap_2$ represents the intersection of R_0 and R_3. Furthermore, let ζ_i be the list of intersecting models under $\cap_i$. To estimate such a mask, we first compute the pixelwise ML estimate according to Equation 4.26. We refine the estimate by employing a voting scheme for each $\cap_i$:

$$\Xi(\cap_i) = \underset{j \in \zeta_i}{\text{argmax}}(\text{count}_j(\Psi(\cap_i))). \tag{4.28}$$

Figure 4-5 shows examples of both the maximum likelihood estimate and the majority-vote estimate of visibility. Of course, there are more sophisticated methods of segmenting the imagery than the methods suggested above (see, e.g. [51, 26]). However, we demonstrate that the above are sufficient for occlusion reasoning under several diverse conditions. These include vehicle tracking in a far-field surveillance setting, described in Section 5.3, and tracking groups of people in a complex, indoor environment, described in Section 5.5.3.

One consideration with these methods of determining visibility masks is that they require a per-pixel computation of likelihoods under the appearance models, which implies pixelwise independence of the shape-free appearance parameters. This suggests a structure to the covariance in appearance estimate that may be restrictive. In practice, however, this is an oft-used assumption, for reasons of speed and computational efficiency. One way to avoid having to compute likelihoods on a pixelwise basis is to estimate the visibility masks in an expectation maximization framework across the image as a whole. Such methods are not considered here, however, as we have found the methods suggested above sufficient for tracking, the results of which are described in Chapter 5.

When part or all of a region R is located outside the boundaries of the imagery, or when part of all of R is occluded, then measurements of appearance for those unobservable locations $H \subseteq R$ will be unavailable. The lack of observations in those regions H requires

slight changes to the detection and update steps described previously. The summation over locations $u \in R_\alpha$ in Equations 4.18 and 4.21 is now over all predicted visible locations within R_α. Likewise, the update occurs only over those locations currently predicted visible.

The incorporation of occlusion reasoning into the tracking process described above allows the tracker to maintain estimates of shape and appearance for the various regions during periods in which part or all of the regions are hidden from view. When those parts become visible again, they again become part of the tracking process, through their inclusion in the detection and update processes.

4.7 Algorithm

The full procedure for tracking with the model described in this chapter is as described in Algorithm 1.

4.8 Summary

In this chapter we introduced an appearance model capable of representing dynamic regions of shape and texture and described how it can be used within a recursive estimation framework for the purposes of online visual tracking.

The appearance model represents shape by estimating the parameters of a piecewise affine mesh. The parameters are the x- and y- locations of each vertex of the mesh within the imagery. The complete shape state that is estimated consists of those mesh parameters and their velocities.

The appearance model represents appearance in a shape-free context. A shape-free context is one in which shape considerations have been normalized out. Though the shape-free context suggested here is the initial shape at $t = 0$, a method of reparametrizing the shape-free context is also described.

A method of detecting instances of tracked regions within the imagery, based on a cost minimizing, gradient-descent registration framework has also been described. The detector

Algorithm 1

Given an image sequence $I_1, ...I_T$, a filter basis ϕ, and a set of models $m_{1,1}, ...m_{M,1}$ with appearance, shape, and visibility masks initialized at $t = 1$.

1: **for** $t = 1$ to T **do**
2: **for** $i = 1$ to M **do**
3: predict appearance and shape of m_i according to Equations 4.14-4.17. This involves finding the positions of the mesh vertices of each piecewise affine mesh by forward predicting the shape estimates according to the shape dynamic. Also, the appearance of the region within the shape-free context is computed similarly.
4: find instance of m_i in I_t according to Equations 4.18,4.20, 4.21, taking the predicted visibility into consideration. This involves fitting model to imagery in a registration-based approach. The predicted visibility of each location within the appearance model determines which pixels are used during the detection process.
5: **end for**
6: update visibility masks according to either Equation 4.26 or 4.28. This involves determining the areas of overlap between detected regions, warping each region onto each model's shape free context, and computing likelihoods. Each pixel's ML visibility mask is determined to be the model where that pixel's appearance is most likely. If using the majority-vote estimation method, then for each area in which two or more models overlap, the model which accounts for the largest portion of visibility for that area is responsible for the entire area.
7: **for** $i = 1$ to M **do**
8: update m_i according to Equations 4.22-4.25, taking the estimated visibility into consideration. This involves incorporating the measurement acquired during the detection process into the estimates of shape and appearance maintained for each tracked region. The visibility masks jointly estimated over all tracked regions is used to indicate which pixels in each measurement are valid and can be used to update each model.
9: **end for**
10: **end for**

takes shape priors into consideration to provide robustness to outliers, visual distractors, and to provide the mesh with rigidity in the absence of visual data. A lack of visual imagery can occur for two reasons: the mesh region not being fully contained within the domain of the imagery, and due to occlusion with other tracked regions.

The ability to reason about occlusion is integral to the formulation of the appearance model described in this chapter. We describe occlusion reasoning as an M-way classification problem: given M models and the current image I, determine which locations within I are best described by which models. We describe two simple, though effective, methods of determining these visibility masks: via maximum likelihood estimation, and via a majority-

vote scheme. However, by posing occlusion reasoning in this context, it is straightforward to utilize more sophisticated classifiers.

In the next chapter, we demonstrate the utility of such a tracking system by tracking various objects and regions under a variety of conditions. Some of those conditions include indoor and outdoor scenes, tracking through occlusion, far-field and mid-field views, tracking of crowds, and tracking of geophysical data.

Chapter 5

Experimental Evaluation

In this chapter, we show many examples of tracking with the model described in chapter 4. Such experiments highlight the utility of the model in tracking diverse phenomena, many exhibiting complex deformations in both shape and appearance. Such phenomena include meteorological data, groups of people, rigid bodies such as cars undergoing pose changes, and even geophysical seismic imagery. We show how the models maintained over the course of the tracked sequence can be used to synthesize examples of the tracked region and how they can be used for image denoising, to arrive at an estimate of the shape and appearance of the object that is more accurate than any observed instance. We also verify the accuracy of the approach by using the resultant tracks in an application of anomaly detection, which is then compared against expertly annotated data. We conclude with a discussion about the time complexity of the approach.

5.1 Hurricane Tracking

The tracking of meteorological data such as hurricanes and other storm systems provides good examples of deforming, nonstationary dynamic textures. The movements of hurricanes are often difficult to predict, both in the short term, where one would like to get as accurate a measure of the storm's size and shape as possible, and in the long term, in order to prepare areas for a possible landfall. Hurricanes also vary significantly in appearance over the life of the system, with features changing continuously, such as the size and

position of the eye, and the size and distinctiveness of the individual rainbands.

As such, dynamic models of appearance and their recent extensions [18, 17] would have difficulty modeling the entire extent of variability in shape and appearance that such a region can undergo. This is due to the fact that the changes to appearance and shape do not necessarily repeat, and so do not remain stationary. We model the variability in shape and appearance by maintaining and reestimating the model online, at each time t. Because we update the model, it only has to be sufficiently expressive to predict the appearance and shape of the hurricane for a short duration.

Figure 5-1 shows the result of tracking a hurricane with the joint shape and appearance model described in chapter 4. We manually initialized the model over the hurricane in the first frame and tracked it through the 40-frame sequence. For this example, we do not process the imagery through a filter bank but operate directly on the RGB data. We are able to maintain the track throughout the sequence because we are able to reestimate the state after each image in an online framework. In this example, the uneven rotation and expansion of the hurricane is sufficiently modeled to maintain its track. As shown in Figure 5-1, this results in shape-free measurements of appearance in which the deformations and rotation of the hurricane are compensated for. Thus, within the shape-free context, the hurricane is largely stationary, not exhibiting the rotation characteristic to all such storm systems nor the more unpredictable, local deformations that can be seen. Normalizing for such changes in shape allows the appearance to be modeled in an environment free from such considerations.

This results in the accurate modeling of the eye of the storm, which disappears halfway through the tracked sequence, and of the bands, which vary in size and distinctiveness throughout the sequence. Furthermore, the bands are asymmetric, appearing on only one side of the hurricane. As the hurricane rotates and deforms, so too do the rainbands. However, because we model the appearance of the bands within the shape-free context, and because we accurately track the shape and position of the hurricane, the bands appear stationary within the shape-free context. The model also remains robust to the stationary, visual artifacts caused by the overlay of state boundaries on the imagery.

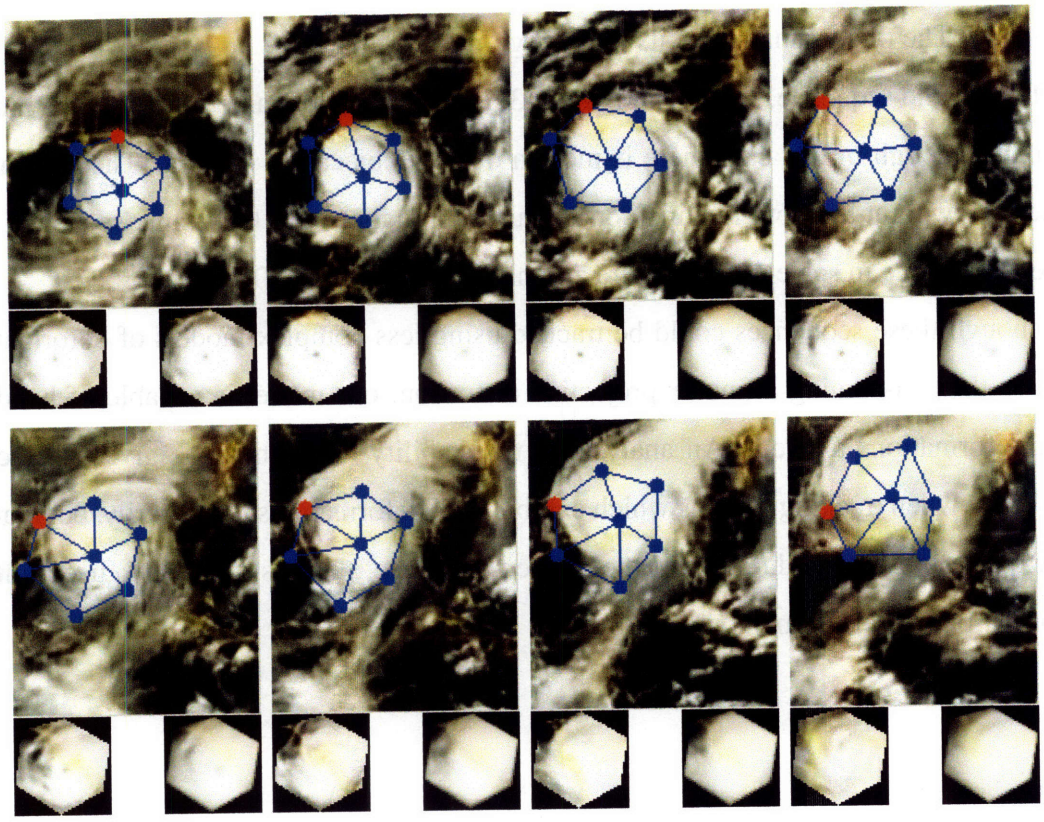


Figure 5-1: Hurricane tracking in the Gulf of Mexico. Though the appearance and shape of the hurricane changes significantly during this short sequence, the model is expressive enough to maintain its track. One vertex was colored differently to highlight the rotational aspect of the system. Bottom left: shape-free appearance. Bottom right: estimated appearance.

5.2 Tracking with PTZ Cameras

Tracking within a panning, tilting, and zooming camera presents a challenge due to the fact that there are two sources of motion acting on the objects within the imagery: the motion of the objects themselves and the independent motion of the camera. We show results on two such sequences. Both sequences were initialized manually and operate on the RGB imagery. The first, shown in Figure 5-2, is a 100-frame sequence of a marching band on a football field. The predominant motion within this sequence is the panning, tilting, and zooming action of the camera. The scene is relatively stable until around frame 30, when the camera begins to move and zoom out, to get a better view of the field. The zooming continues for several frames, resulting in a view of the group that is half the size that it was

originally. The tracker maintains its target region throughout.

The second sequence, shown in Figure 5-3, is a similar 100-frame sequence of a marching band, though in this case there is motion of both object and camera. The region and camera viewpoint are both moving continuously throughout the 100-frame sequence, and there is a change in scale as the camera zooms in around frame 50.

Though these sequences could be tracked using less complex models of deformation, such as those assuming affine or projective distortion, our model is capable of handling such deformations as well. An analysis of the stability of the mesh is given in Section 5.8. Thus, when tracking regions which exhibit such deformations temporarily, but that are not expected to for the duration of the tracked sequence, there is no need to use multiple deformation models to maintain the track.

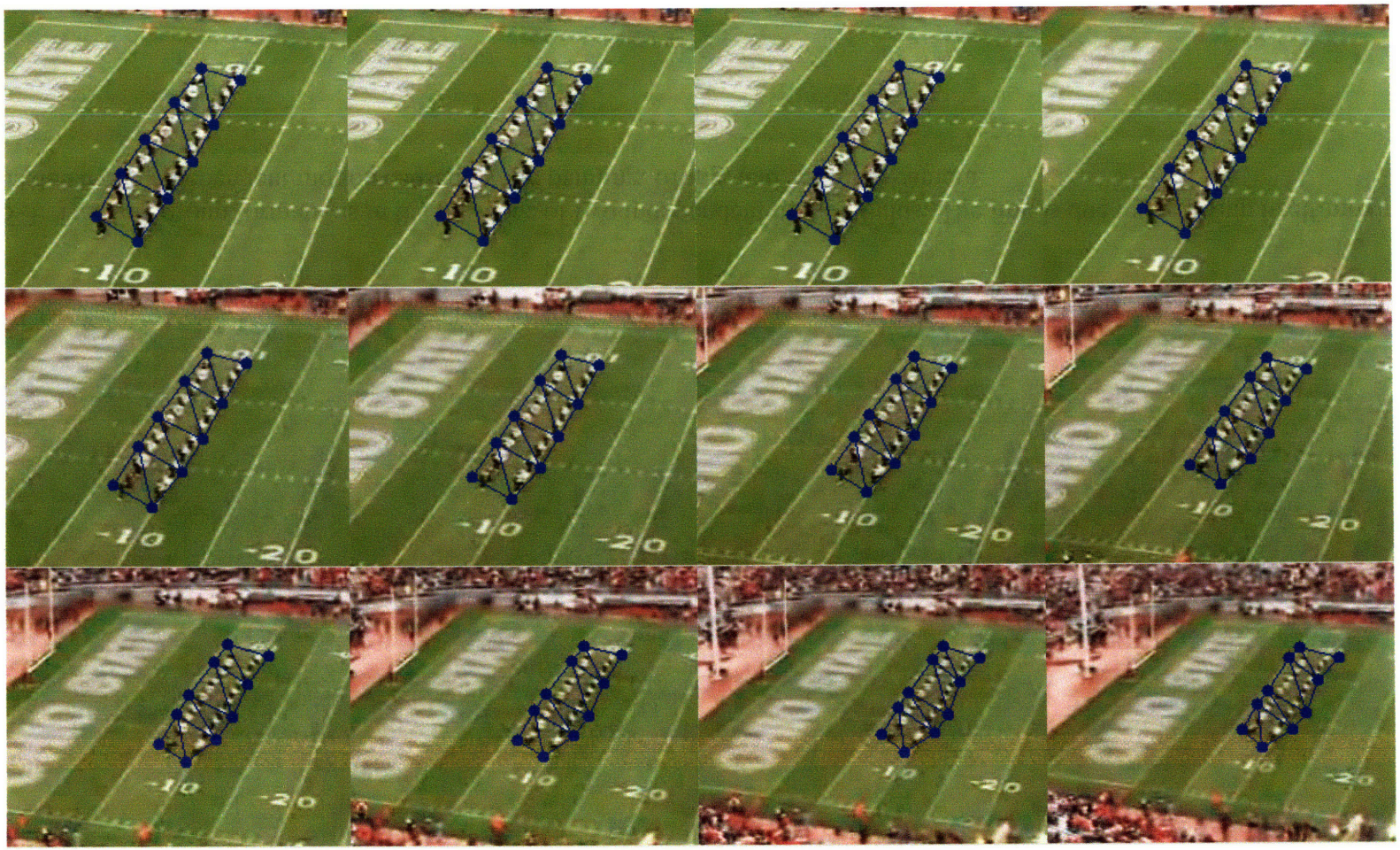


Figure 5-2: Tracking group formations within a panning, tilting, and zooming camera.

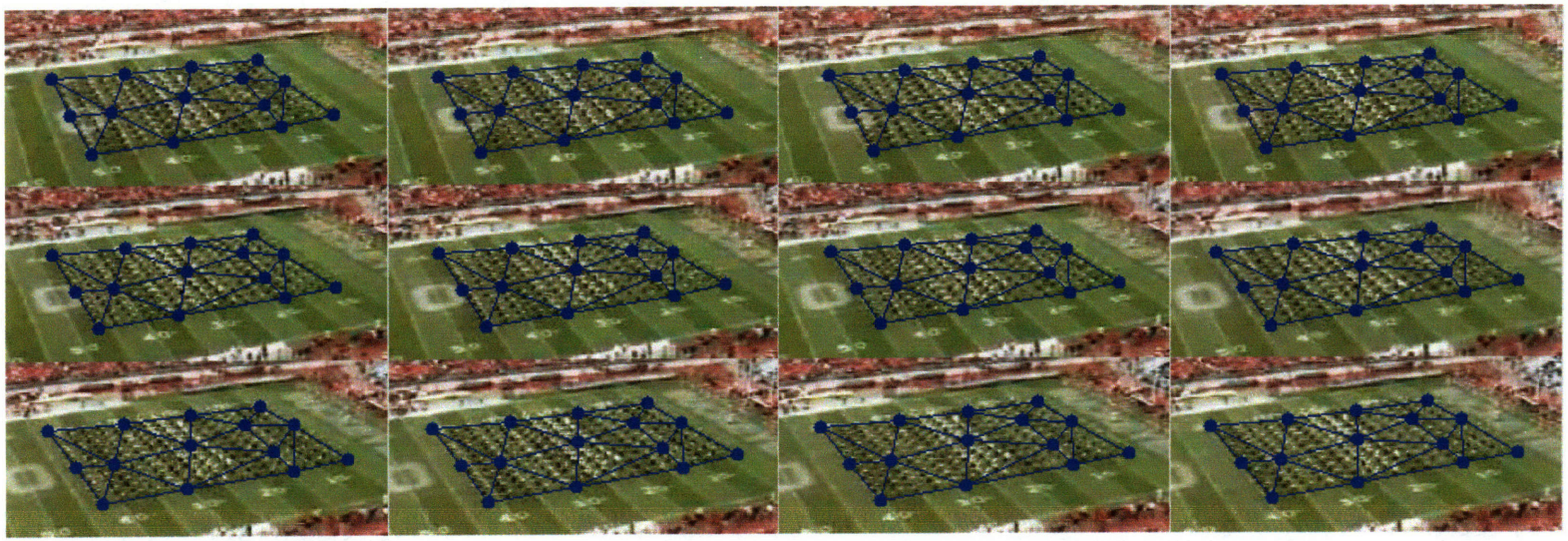


Figure 5-3: Tracking group formations in the presence of two independent motion fields. The first is due to the group itself moving. The second is due to the independent motion caused by the panning, tilting, and zooming camera.

5.3 Vehicle Tracking through Occlusion

The tracking of cars and other motor vehicles is a typical scenario used to showcase many tracking systems due to the constrained environments in which cars travel. However, for intersections as busy as the one shown in Figure 5-4, many approaches to tracking would fail. For example, blob trackers would have a difficult time with such a scene, as there are many opportunities for occlusions, resulting in many merged and split tracks. It would also be difficult to track this scene with layered models, which solve the problem of occlusion by estimating full depth-based orderings of the various objects. However, because they require prior knowledge about the numbers of expected layers and because objects come and go frequently, such approaches cannot be readily applied to this scenario.

Our approach to tracking through occlusion rests on the predictive capability of the joint shape and appearance model. For this sequence, we use the method of determining occlusion boundaries given in Equation 4.26. This produces estimates of model visibility on a per-pixel basis. Instead of modeling the entire z-order of the meshes explicitly, this likelihood test is sufficient to continue tracking and maintain our models through the occluded region. Indeed, deformations in the unoccluded regions of the "blue" car carry over into the area of occlusion and continue to evolve the model in those areas. Also, because the update step is performed only over those pixels currently visible, the models do not get corrupted by the occluding objects.

Figure 5-4 shows the results of the tracked sequence. Initially, when the cars are well separated and we have unoccluded views of them, the appearance models and observations of shape-free appearance are similar. This is because the appearance of each car within the shape-free context remains largely unchanged. During this sequence, the red car gets occluded by a light pole that is in the foreground. Because the car remains stationary in its shape-free context, the truly stationary light pole appears to move within that same context. This transient object has a subtle effect on the appearance model of the car, due to it being incorporated into the model during the update step. This manifests as a slight ghosting effect that is quickly eliminated as the car continues to move past the light pole.

Also of interest are those pixels incorrectly labeled as visible (and not visible) as a

result of occlusion reasoning. These follow the line of the windows within the red car. The windows appear similarly colored to the occluded car, both of which appear dark, and so the per-pixel likelihood test is less decisive in those areas, resulting in noisy segmentations. However, because of the prevalence of evidence elsewhere, the tracker remains robust to such estimation error.

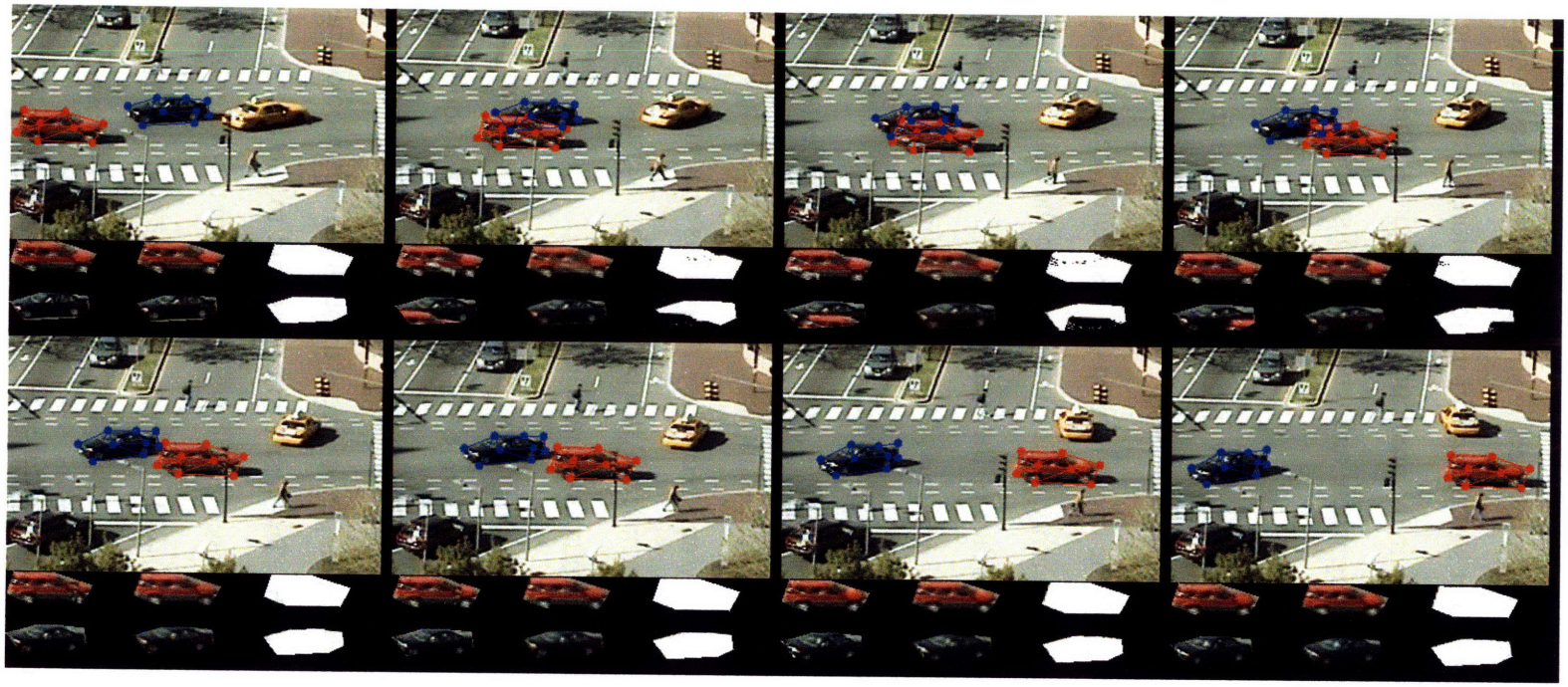


Figure 5-4: Tracking through occlusion. A probabilistic appearance model is maintained for each tracked mesh, enabling simple segmentation-based occlusion reasoning. Only visible areas of the meshes are used to track and update the joint state space over deformations and appearance. Top: tracked cars. Bottom left: shape-free measurements. Bottom center: estimated appearance. Bottom right: occlusion masks.

5.4 Seismic Imagery

Imagery of geological data are studied by geologists and geophysicists in order to better understand the processes that affect soil composition, formation, and geologic evolution. Insights into such processes can help in the prediction of earthquakes and volcanic activity, for example, by better understanding where and when faults, fissures, and other defects in the soil layers arise.

The imagery is obtained using an ultra low frequency sonar device aimed at the ground and a microphone array to listen for the reflections. The device is located on a boat which follows a predefined path through the ocean to map out a specific area of the sea bed and soil layers below. The microphone array, which is tethered to the boat, is situated several meters behind it, floating on the surface of the water. Figure 5-5 shows an illustration of the process of acquiring the seismic imagery. At each location, the positions of the boat and tethered microphones are carefully noted, and a "ping" is sent down into the water and through several kilometers of the earth below. The sound will be reflected at the interface of different soil layers, due to differences in density. Some time later, as a function of soil density and depth, the primary and secondary reflections will make their way up to the microphone array and will be recorded. After collecting the measurements, the boat is moved to the next location and the process is repeated.

What results from this data collection process is a volume of seismic imagery consisting of several million volumetric pixels or voxels, the three dimensional equivalent of pixels. The seismic volume we analyze has dimensions $321 \times 326 \times 1751$, containing approximately 183M voxels. Figure 5-6 shows examples of slices through the seismic volume. The axes are labeled track, bin, and depth. The first two correspond to the boat position on the surface of the water and the last corresponds to the depth within the water or soil layers.

As can be seen from the horizontal slice (Figure 5-6, left), there are two dominant structures present in the imagery. There is a circular structure to the lower-right which is surrounded by oriented, wavy bands of texture. The banded regions are caused by horizons intersecting the horizontal plane. Horizons represent a single layer of near uniform soil composition, such as sand, shale, or clay. These layers can be seen in Figure 5-6 (right).

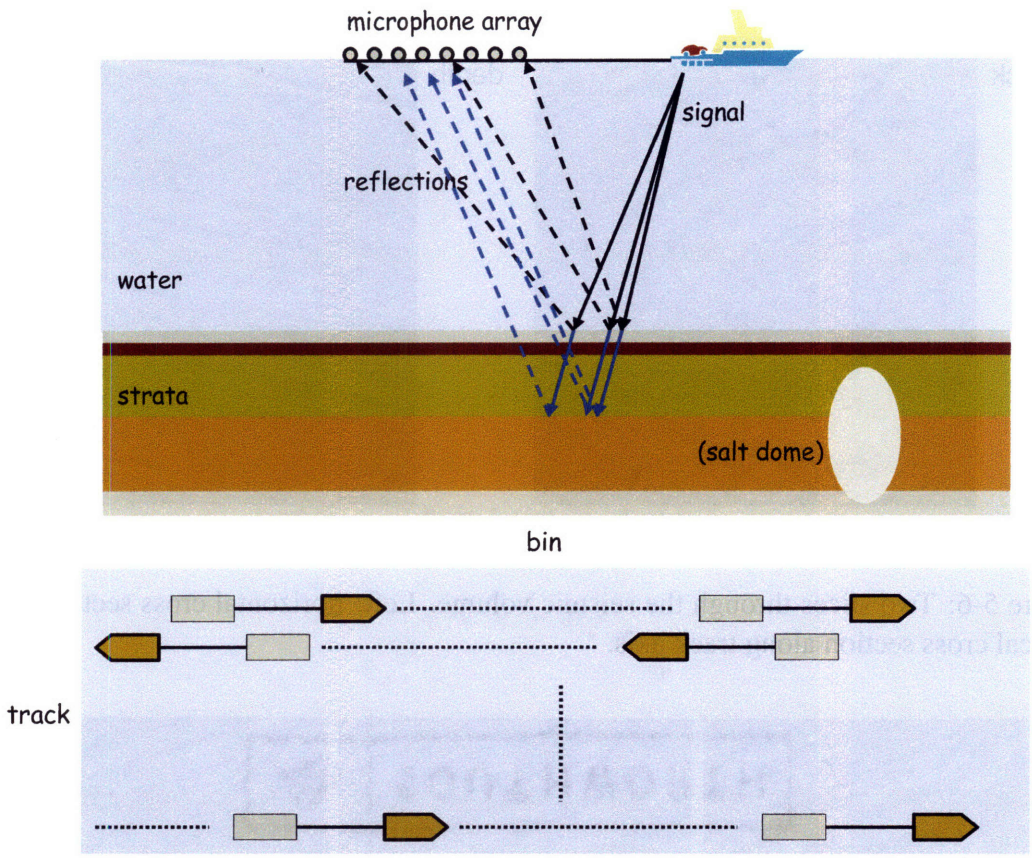


Figure 5-5: Acquisition of seismic imagery. A microphone array is tethered to a boat containing a sonar device. At each location, a ping is sent down and the reflections of that signal are collected by the microphone array. The boat moves on the surface in a regular grid pattern, sending signals at predefined intervals. The data are then integrated into a seismic volume.

Despite their name, they are not, in fact, perfectly horizontal. Thus, the layers intersect the horizontal at certain points in depth. When viewed from horizonal cross sections, the layers can be seen to "pierce" through, giving rise to the banded region we see. The circular region in the lower-right is a salt dome. Salt domes form when salt permeates the various layers of the soil, displacing and compressing the surrounding soil to make room for the salt.

A challenge with analyzing seismic imagery is the sheer size of it. A typical survey can cover several square kilometers of surface area and extend several kilometers below ground. It can take geologists on the order of several months to annotate a small fraction of the total data. A typical task left to geologists is to locate and label the various soil

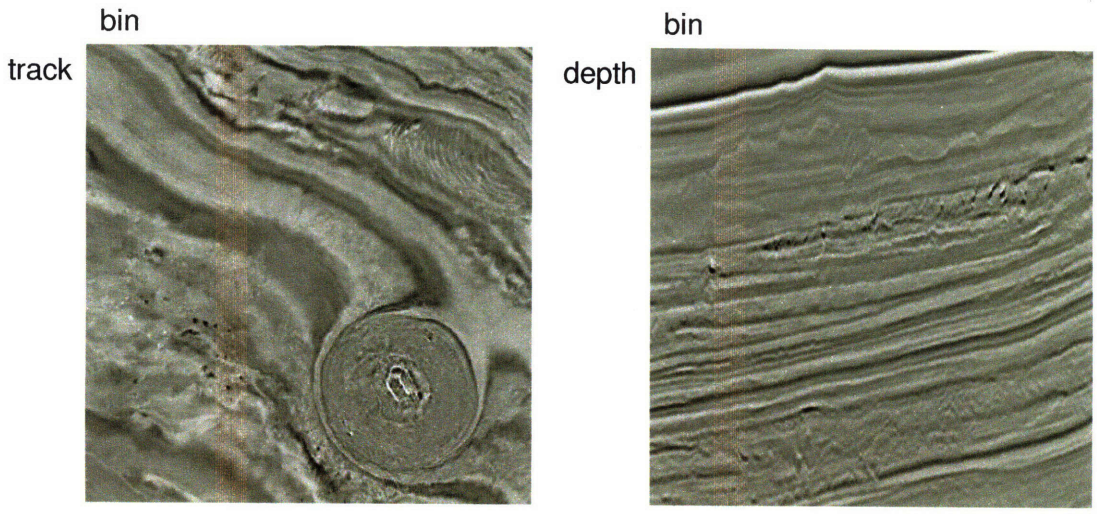


Figure 5-6: Two slices through the seismic volume. Left: horizontal cross section. Right: vertical cross section along track axis.

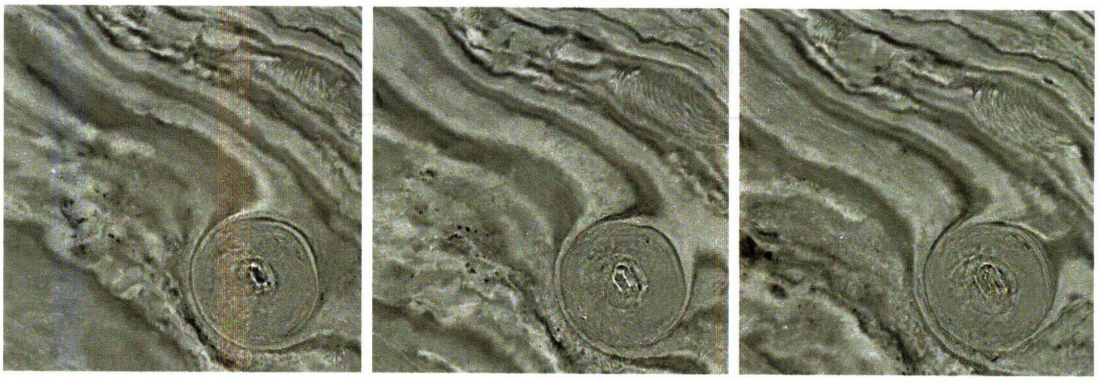


Figure 5-7: When the depth axis is treated as time, the intersection of soil layers, or horizons, with horizontal cross sections appear to move.

layers, or horizons. Finding these horizons is challenging because they are not uniform in appearance, shape, or location. Thus, correlation-based trackers often fail to successfully track the extent of the horizon.

The approach we take to tracking the horizons is based on a different interpretation of the seismic volume. We note that if the depth axis is treated as a time axis instead of a third spatial axis, then the difference in position of the intersection between soil layer and horizontal cross section manifests as apparent motion. Figure 5-7 illustrates the effect. The motion is not uniform, however. As can be seen in Figure 5-6 (right), the horizons do not

maintain the same incline with respect to the horizontal. Furthermore, the relative distance between the various layers change. Under horizontal cross sections, and treating depth as time, this manifests as motion of the textures comprising the intersection of horizon with cross section that move at different speeds. This results in a nonuniform deformation of the horizons. The appearance of the layers within the horizontal cross sections changes over time (depth) as well. The further down in depth one images, the less detailed the image becomes. Also, different geologic considerations lead to the textured appearance of single layers to change with depth, as well.

Thus, horizons can be aptly described in terms of regions undergoing dynamic, nonuniform changes to both shape and texture. To facilitate the tasks of automatically annotating the horizons, we track these deformations and show how it leads to tracking of the horizons.

5.4.1 Texture Basis

To facilitate the task of automatic annotation of seismic imagery via tracking, we first define a basis in which to represent the textured appearance of the layers. Since the horizons exhibit a coarse-scale, oriented appearance, we analyze the imagery by filtering through an oriented multiscale wavelet basis, resulting in a steerable pyramid [25]. The pyramidal decomposition we use yields a four-orientation five-scale pyramid. Figure 5-8 shows such a decomposition into its various scales and orientations.

For efficient use of such a representation, it is helpful to upsample the coarser scales of the pyramid, so that every location in the image can be described by a feature vector of length 20, composed of filter responses over four orientations and five scales. Because of the dimensionality of the data, and because we expect that such coarse, oriented features of the horizon regions will have filtered responses in only certain scale/orientation combinations, we define a region in a single image from which to learn a low dimensional subspace within the larger 20-dimensional texture space where such features may naturally cluster. Such initialization can easily be done by an expert operator. The region we choose is shown in Figure 5-9.

The subspace is based on principal component analysis. Let $F = \{f_1 ... f_P\}$ be the

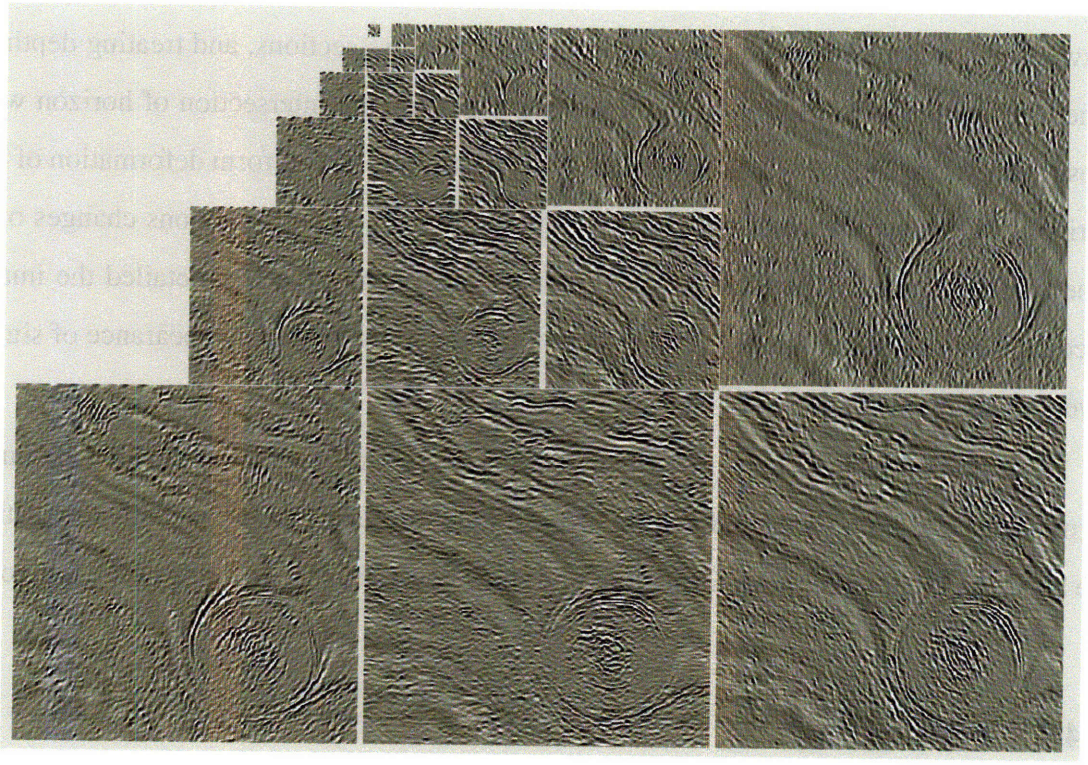


Figure 5-8: Pyramidal decomposition of the seismic imagery using four orientations across five scales. These are shown as five sets of four equally sized images. The topmost image of the pyramid represents the low-pass residual, which is unused in our experiments. The horizon region exhibits "preferred" scales and orientations that correspond to the natural scale and orientation of their textured appearance. These manifest as very bold regions, indicating strong responses at those scales and orientations. In contrast, the circular region in the lower-right (salt dome) shows no such preference for any scale or orientation.

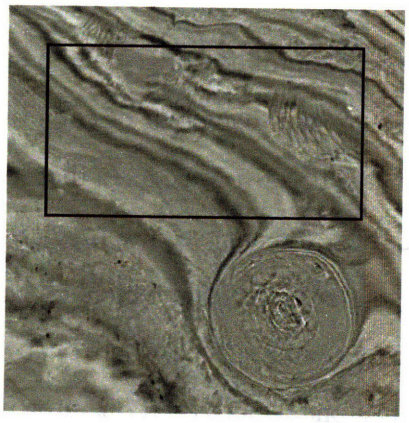


Figure 5-9: Region used to learn texture subspace.

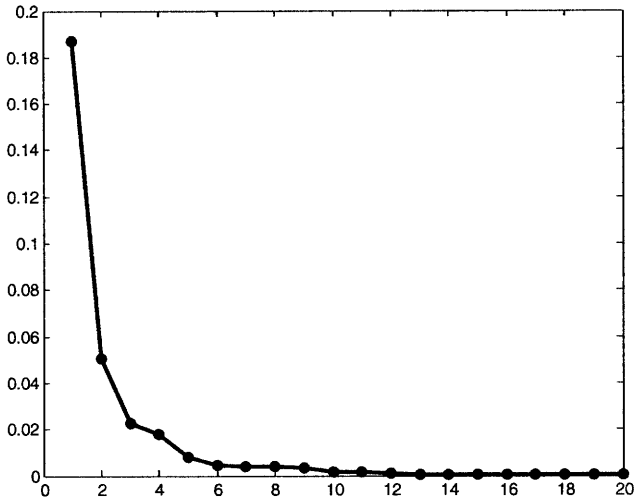


Figure 5-10: Sorted list of eigenvalues from data matrix consisting of feature vectors derived from a steerable pyramid decomposition of horizontal slices of the seismic volume.

set of feature vectors at locations $1...P$. In the results of this section, each f_i is a vector of length 20, representing filter responses comprising four orientations across five scales. Let their mean appearance be $\Psi = \frac{1}{P}\sum_i f_i$ and let the 0-centered data matrix $A = \{(f_1 - \Psi), (f_2 - \Psi), ..., (f_P - \Psi)\}$. The principal components can then be derived by

$$AA^T v_i = \lambda_i v_i, \tag{5.1}$$

where v_i,λ_i represent the eigenvectors and eigenvalues of the data covariance matrix AA^T. The size of the eigenvalues indicates the relative spread of the data along the corresponding eigenvectors. Figure 5-10 shows a plot of the sorted eigenvalues. Most of the variance in the data is retained in the first few eigenvectors. We choose as a subspace the four largest eigenvectors, which results in a retention of approximately 90% of the variance. To project a feature vector f onto the subspace, we have

$$\epsilon(i) = v_i^T(f - \Psi), \tag{5.2}$$

where $\epsilon(i)$ is the projection of $f - \Psi$ onto the ith eigenvector. For the subspace used here, $i = 1..4$. The largest three components of such a projection are shown in Figure 5-11. The predominance of "primary" colors red, green, and blue indicate that those regions are

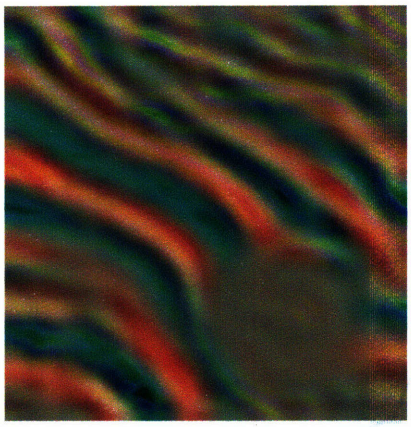


Figure 5-11: Projection of steerable pyramid analysis onto largest three eigenvectors of PCA subspace.

described primarily by the projection onto a single eigenvector. The repeating nature of the colors is indicative of the fact that the soil layers lie one atop the other.

To motivate the use of this texture basis, consider the example shown in Figure 5-12. Because the learned PCA subspace well models the horizon region, the tracking based on it is more accurate since the texture space more accurately represents the horizon data at each location than does raw intensity values. This can be seen clearly in the lower-left corner, as the texture-based tracker models the deformation of the large lighter-toned band.

5.4.2 Horizon Tracking

The horizon region was tracked for 100 frames. As shown in Figure 5-13, a square, tesselated grid was initialized such that the top portion of the grid started off the available imagery. Furthermore, for the purposes of tracking, the salt dome was masked out as unavailable or invalid imagery, an operation that an expert could easily perform. Such user assistance is regularly performed in the interpretation of geologic data, as it is often straightforward to delineate the boundaries of such regions, but difficult to interpret the data within a particular region. In the case described here, for example, it is straightforward to separate the salt from the horizon data, but difficult to find the specific layers of soil within the horizon region.

One thing to note is that the corner elements of the mesh deform more than the interior

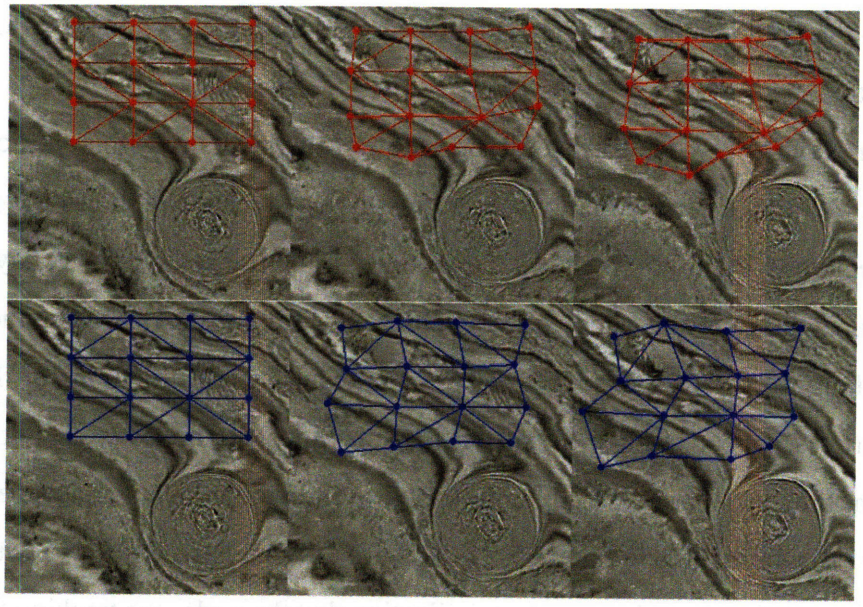


Figure 5-12: Tracking with intensity (top) vs. learned texture basis (bottom). Tracking within the texture basis results in more accurate estimates of the deformations the horizon region undergoes.

elements do. This is because there are less constraints on the positions that the nodes comprising those elements can take. The corner nodes, which are responsible for the affine deformation of only a single element, are particularly vulnerable to this lack of constraint. However, the upper-right node remains in the same relative position with respect to the oriented bands directly below it throughout the sequence, indicating a successful track. Furthermore, because it is influenced by the positions of the other nodes via the spring prior on shape, it is able to track laterally, in a direction parallel to that of the surrounding texture. Due to the aperture problem, in which local estimates of motion can only be derived in a direction orthogonal to the curve, this is not solvable by models that look at local texture only.

The ability of the model to take into consideration regional information also makes the registration and tracking process robust to structured noise. For example, at the beginning of the track in the upper-left of the region, there is a large, stationary sensor artifact present. Over the 100-frame sequence, the artifact attenuates and, eventually, disappears entirely. Throughout, the tracking of the deformation remains robust to this stationary fea-

ture, due to the predominance of the evidence surrounding it suggesting a different model of deformation.

Another way to view the tracked deformation of the layers is within vertical slices of the seismic volume, a view commonly used by geologists since it shows the horizons in a more direct way. Figure 5-14 shows the vertices of the tracked mesh overlayed on such a view. Here, the different slices do not represent different depths, but rather different positions along the "track" axis. The result of tracking the deformations of the layers is that the vertices appear to slide along the layers. Since the nodes were initialized in a grid pattern, they do not necessarily correspond to individual layers and therefore do not necessarily fall within them in Figure 5-14. However, their relative positions with respect to the layers around them remains constant, indicating that the nodes are tracking the horizons successfully. These tracks can be used by geologists to facilitate the tasks of detection and segmentation of the horizons, especially in the cases where simple correlation approaches fail due to a lack of local information, such as when layers temporarily "pinch" together or are corrupted by sensor noise.

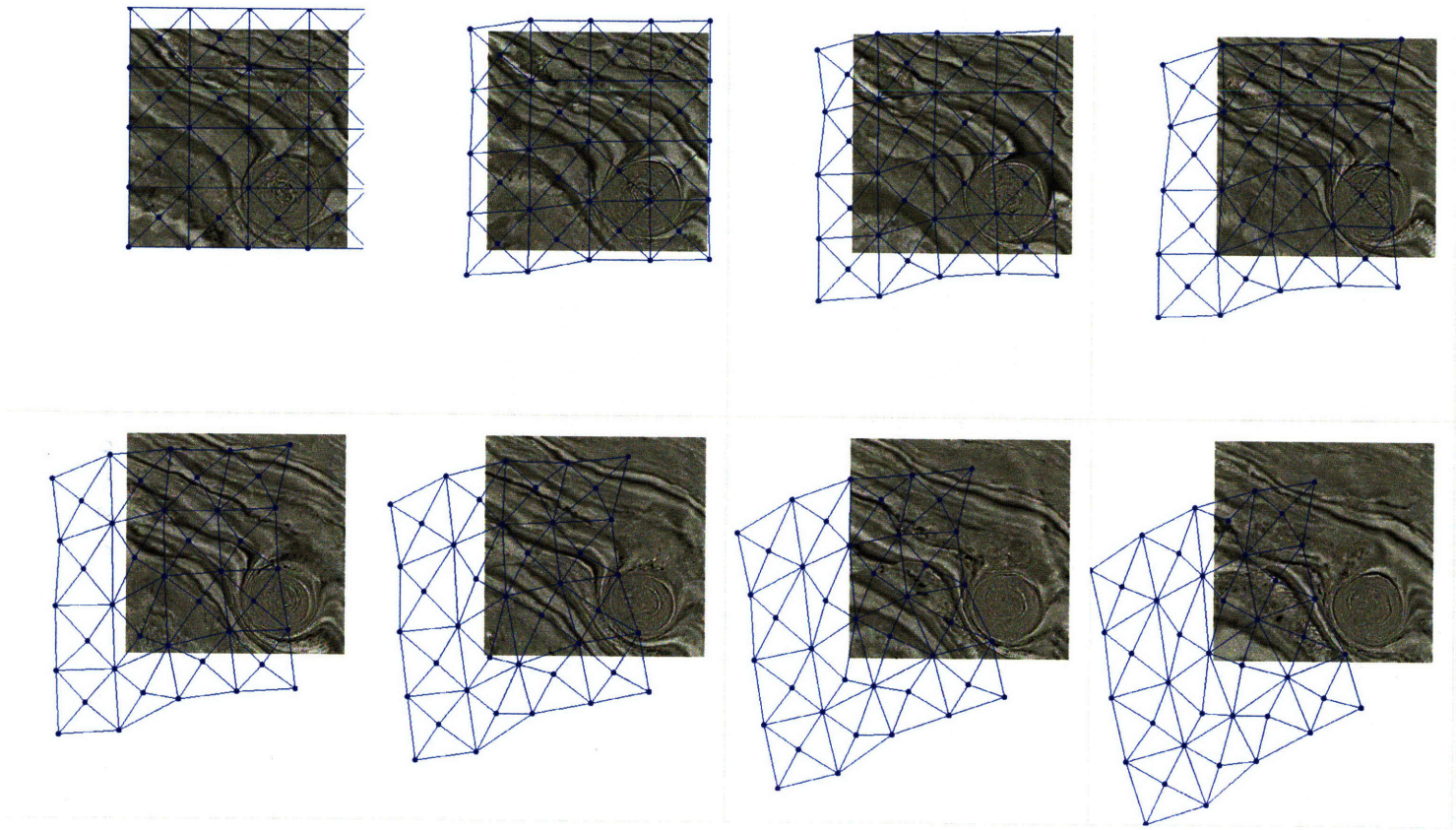


Figure 5-13: Horizon tracking over a 100-frame sequence in a seismic volume. The depth axis is treated as a time axis, which allows the intersections of horizons with horizontal cross sections to be treated as deformations of the region.

5.4.3 Applications

Two applications of tracking the horizon region are considered. The first is an application in synthesis: to predict the appearance of the horizon region outside the bounds of available imagery, based on the appearance of the region within and the model of deformation. Also, by taking the appearance and shape deformation into consideration, we make a prediction about what the horizon region would have looked like had the salt dome not interfered with the evolution of the soil layers.

The second application is of anomaly detection. By analyzing the shape-free appearance of the horizon region, we are able to highlight areas whose local deformations are not well described by the model of deformation learned over the course of the tracked sequence. Such areas are of interest to geologists, as they indicate potential sources of geologic activity, such as those caused by faults and salt permeation.

Synthesis

To synthesize the appearance and shape of the horizon region, a smoothed appearance model was created for each horizontal slice using the entire tracked sequence of shape and shape-free appearance. This model was then forward warped onto the tracked shape and overlaid over the original imagery. Figure 5-15 shows examples of this synthesis. The synthesis predicts what the appearance of the horizon region would look like if the salt dome was not present. By masking out the salt dome, we treat that region as missing data. Since the deformation takes the horizons "through" that region, we have samples of its appearance both before and after the various horizons enter that unobservable area. From this data we can infer what they would look like within the region.

The synthesis is not perfect, however, due in part to the fact that the influence of the salt dome extends beyond the physical appearance of it. The salt dome has compacted the soil layers around it, creating complex deformations that are not well modeled by our tracked shape deformation. These complex, local deformations enter into the estimate of shape-free appearance and cause the horizons to flex slightly differently within the salt dome region than they do far away from it.

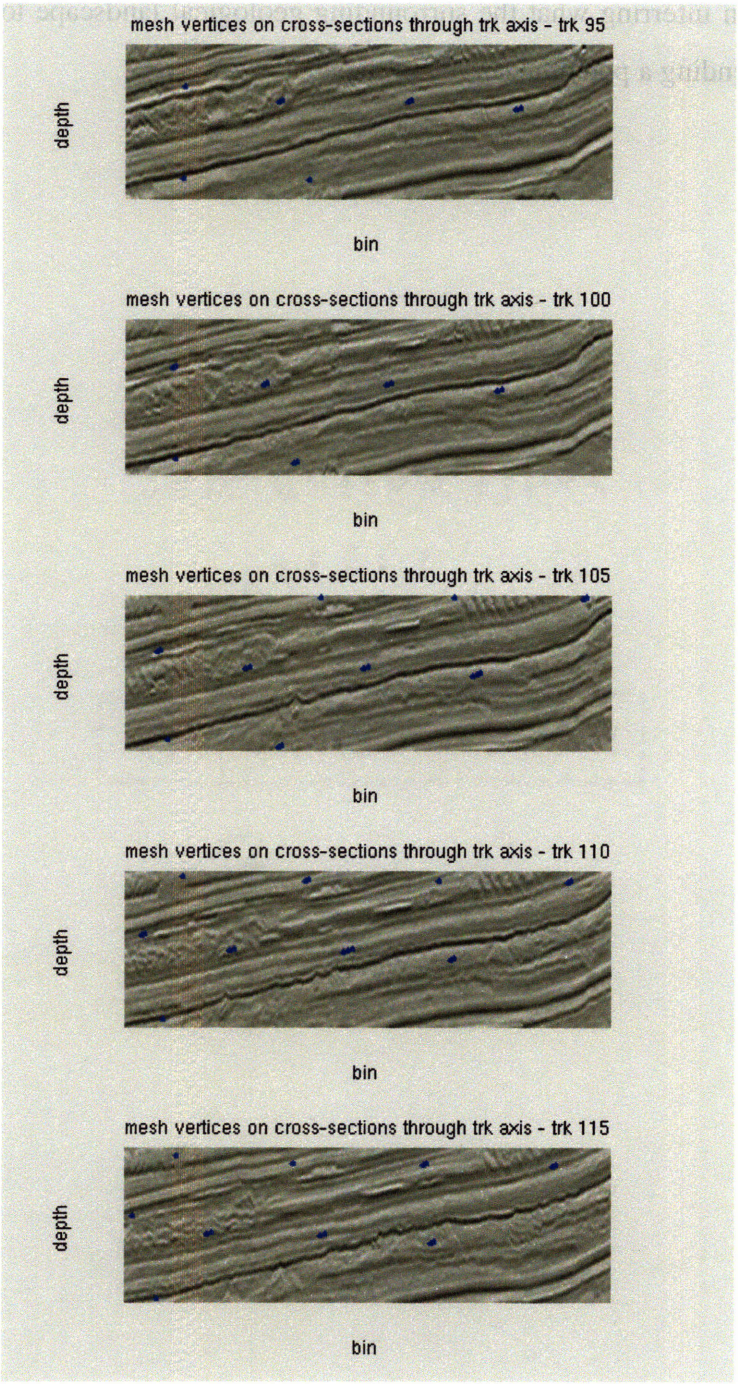


Figure 5-14: Viewing the nodes of the tracked mesh in a vertical cross section shows how the mesh tracks horizons. Because the nodes were not initialized on horizons, they do not show up on them here either. The relative distance to the horizons remains constant, however, indicating a successful track.

The synthesis also extended to regions beyond the domain of available imagery. This may be useful in inferring what the surrounding geological landscape looks like, for the purposes of extending a potential survey of the area.

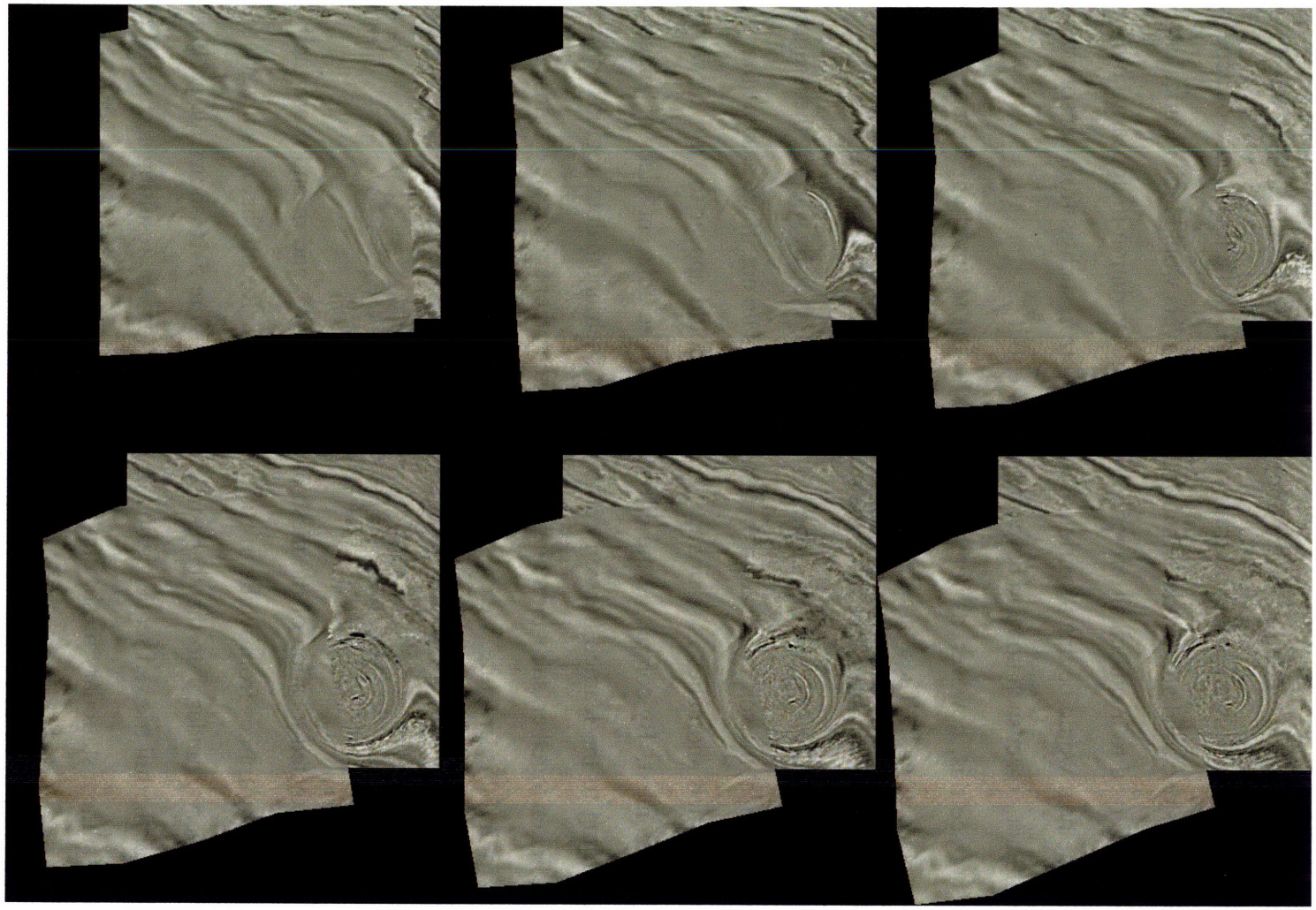


Figure 5-15: Synthesizing the appearance of the horizon region where data is unavailable, using the tracked shape and shape-free appearance learned over the course of the tracked sequence. To show tracking fidelity, the synthesized data is overlaid on the original sequence.

Anomaly Detection

The shape-free appearance of the layers gives us insights into the dynamic geologic processes affecting soil composition. The shape-free appearance models the appearance when the tracked shape deformations are removed from consideration. If the total cause of appearance variation was due solely to that deformation, then the layers would look identical after the shape was warped to a canonical reference frame. Any variation in appearance from one layer to the next would be due to various sources of error, such as from sensor noise, and we would not expect such noise to have any structure to it.

We do not see this, however, as complex changes to the textured appearance of the layers takes place as well. These are due to geologic anomalies such as faults and cracks that do not deform and change shape as do the surrounding, tracked soil layers.

When the deformation of the horizon region is compensated for, by warping the imagery onto the initial shape, the soil layers align, but the anomalies do not. If we assume that the deformation of such anomalies is stationary, then they should appear to "move" within the shape-free context. By computing the difference between adjacent slices of imagery, warped according to the learned horizon deformations, such anomalous regions will be highlighted. Taken together, such anomalies reenforce each other and create a landscape of anomalies. To find this structure, we create an accumulation map, which can be described by:

$$A = \sum_{i=m}^{n} I_i(w(\cdot, p'_i)) - I_{i-1}(w(\cdot, p'_{i-1})), \tag{5.3}$$

where p'_i is the detected location of the mesh at time (depth) i, and m and n are the beginning and ending frames from which to accumulate errors. Recall that w is a warp that brings the imagery I under p' to the shape-free context. Figure 5-16 shows the accumulation map over the full tracked sequence and, for comparison, the resulting accumulation map if you assume a stationary mesh, implying that the layers undergo no such deformations. Because faults occur locally, within a small range of depths, it is often advantageous to limit the accumulation over a subset of the tracked volume. Figure 5-17 shows localized accumulation maps comprised of the first third and the last third of the tracked volume. Also shown are the locations of areas of interest, annotated by a geologist familiar with the

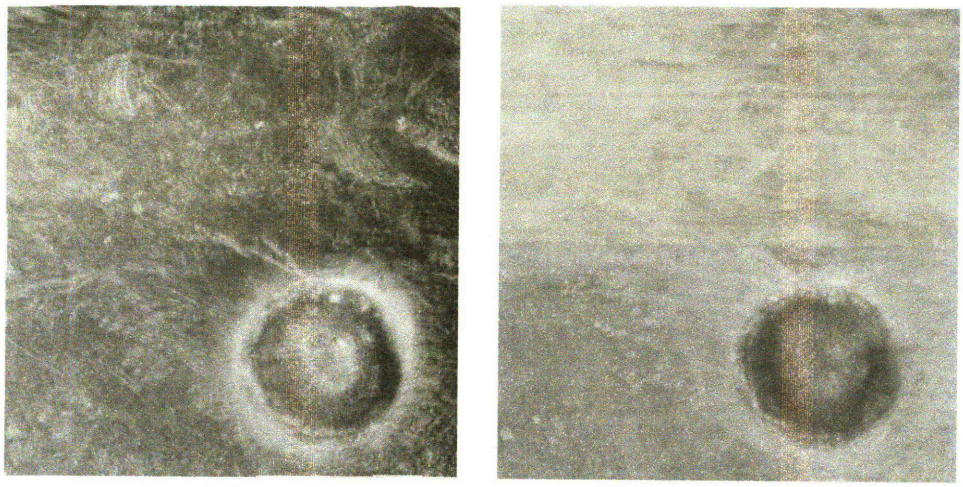


Figure 5-16: Compensating for the deformation of the layers reveals an interesting landscape of geologic anomalies. (a) Accumulation map using models of deformation computed over the tracked sequence and, for comparison, (b) accumulation map computed assuming no deformation.

data. The accumulation maps clearly highlight these areas, and suggest that there are many more such places of interest that have not yet been annotated.

5.5 Group Tracking

Here we consider the problem of tracking groups of people as deformable, dynamic textures. The advantage of tracking groups of people as deformable, textured regions is that, in many circumstances, it eliminates the need to track each person individually, a task which is infeasible in densely populated scenes such as the one shown in Figure 5-18. A common application of tracking data, for example, is to automatically find the sources and sinks of moving objects within the field of view of a stationary camera [61]. Sources and sinks are learned by clustering the origin and destination locations of tracked objects. These locations can be suitably learned by tracking groups of people as well as by tracking each individual, were that possible.

Furthermore, by tracking the groups rather than the individuals, we are robust to conditions that would otherwise confuse other trackers. Such conditions include "outlier" motion

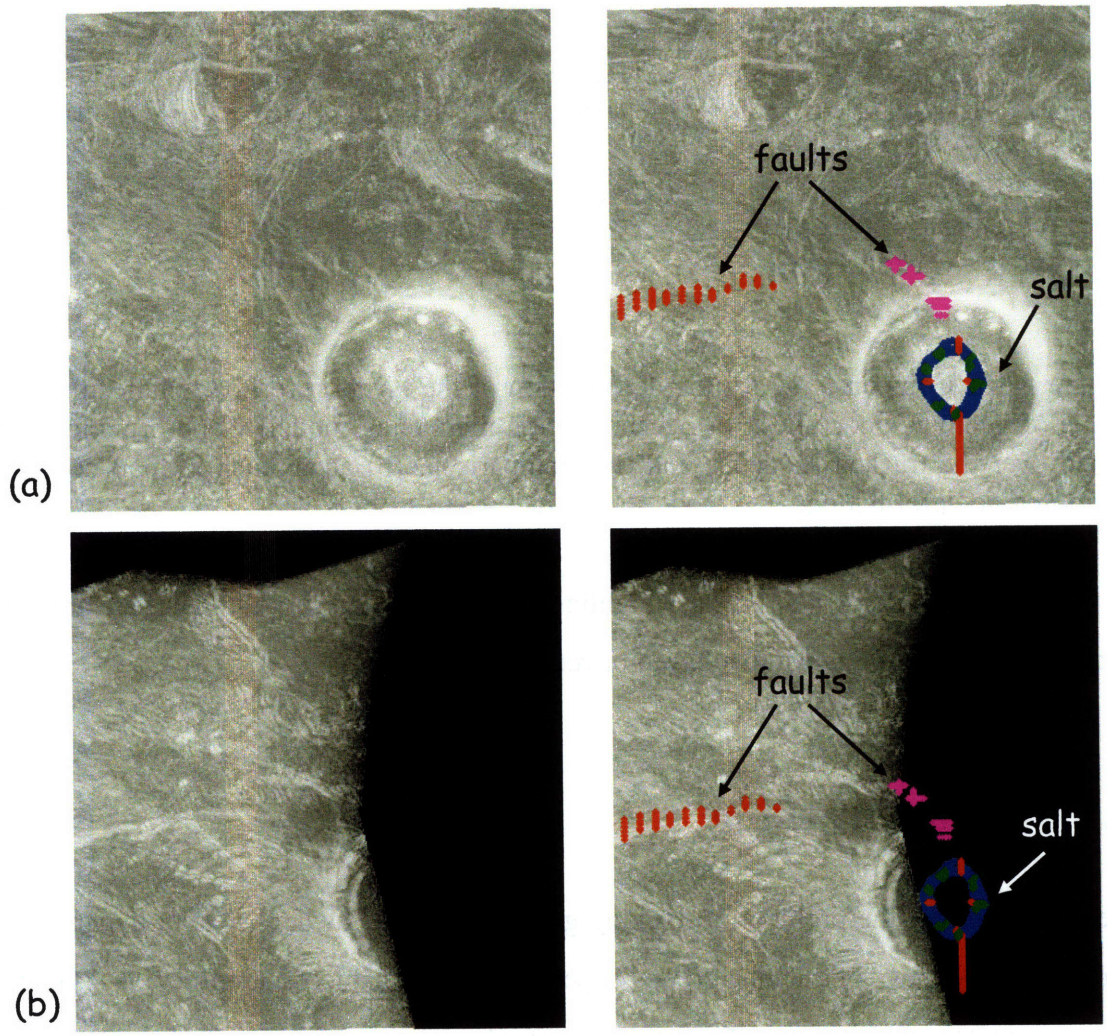


Figure 5-17: Anomaly detection using accumulation maps. Anomalous geologic features are localized in depth and are revealed by computing accumulation maps over portions of the seismic volume. (a) Accumulation map computed using first third of tracked volume. (b) Accumulation map computed using last third. The second column shows an overlay of features annotated by an expert geologist.

Figure 5-18: Tracking groups of people instead of each individual separately enables tracking in situations where it would otherwise be difficult, or impossible. Furthermore, tracking at the group level provides sufficient level of detail to allow many applications in track-based scene analysis.

caused by people not belonging to the group passing through with their own appearance and trajectory unrelated to the appearance and trajectory of the group. Also, we are robust to local displacements, which frequently occur when people's positions with the group change. This is modeled as a change in the textured appearance of the group rather than as a more complex, local deformation. We show results of tracking groups of people as well as cases where we fail to maintain the track, and explain the circumstances that lead to such tracking failures.

5.5.1 Far-field tracking

This sequence consists of a large group of people imaged by a far-field stationary surveillance camera and tracked for 120 frames. The tracked group is surrounded by people whose motions and appearance are contrary to that of the group. Such distractions are a typical source of tracking error. For example, consider the results obtained by using the KLT tracker [44, 58], a flow-based, local feature tracker, to track points within the group of people. The results are shown in Figure 5-19. Of the ten points, only two are tracked successfully. Two get confused due to their proximity to the camera flash. The others get confused due in part to the feature points getting occluded and also due to the proximity

of similar features appearing on individuals moving against the flow of the crowd. This results in their tracks being lost.

Though these types of distractions are a common source of tracking error, we remain robust to them due in part to the shape priors used during detection and by maintaining the textured appearance of the region. The results presented here are shown with enhanced contrast, for illustrative purposes. However, the sequence was tracked without enhancement.

Several interesting events happen over the course of this tracked sequence. The first is that the people within the group frequently change positions. As mentioned previously this induces a change in textured appearance within the shape-free context. It is clear from Figure 5-20 that, even though any single individual within the group may move unpredictably, the group as a whole continues to be tracked.

Also, at frame 14, a person within the group takes a picture, which results in a large lighting artifact caused by the camera's flash. Due to the piecewise affine mesh and the constraints caused by shared vertices, the mesh continues tracking. However, without the constraints, which incorporate regional information about the group dynamics, a tracker would get confused. Figure 5-21 shows the results of such tracking. Here, a single mesh element is tracked, emulating an affine patch tracker. The single mesh element has little additional information to utilize and so does not find that the mesh should move upward, into the area blinded by the camera flash. As a result, it loses its track. The piecewise affine mesh is robust to the flash since all other areas suggest what the proper motion should be.

Figure 5-19: Flow-based tracking fails in such dense and busy indoor environments. Constant occlusions and other visual distractions easily confuse local feature trackers. Only two of the original ten points were tracked successfully through the entire sequence. The last frame shows an overlay of the resultant tracks, with circles in green indicating the start of each track and the red circles indicating the end.

Figure 5-20: Group tracking in densely populated scenes. (a) imagery and detected mesh (b) measurement of shape-free appearance (c) estimated shape-free appearance model

5.5.2 Mid-field Tracking

To illustrate the effectiveness of tracking groups of people rather than individuals, consider the example shown in Figure 5-22. This sequence tracks two regions simultaneously in a mid-field surveillance setting, one representing a group of people and the other a single individual.

As in the scenario from Section 5.5.1, this sequence shows how tracking groups of people who move according to a shared dynamic is more robust than the tracking of individuals in such dense scenes. The tracking of the individual (blue mesh) fails when she is fully occluded by another individual sharing a similar appearance. However, the mesh tracking the group maintains the track even when several untracked individuals interact with the group. This is because, even though several members of the group are occluded at any given time, the prevalence of the motion of the group as a whole still dominates. Compare that with the motion inferred on the mesh tracking the individual. In this case, when she is occluded, there is no prevalence of motion elsewhere to continue detecting the region's shape or appearance accurately, and the track is lost.

5.5.3 Group Tracking Through Occlusion

We have previously demonstrated how to track groups of people using the joint model of shape and appearance suggested here and how tracking groups proved more reliable than tracking individuals in crowded scenes. In this section, we show how simpler models of deformation fail to accurately capture the shape deformations of the group and how this leads to failed tracks. We also add occlusion reasoning to the challenges of group tracking in real-world scenarios by tracking through an area of sustained occlusion.

To illustrate the effect of the articulated, deformable shape model, consider the tracked mesh shown in Figure 5-24. With few points of articulation and few mesh elements, the shape is too rigid to maintain the track. The mesh get easily distracted by the untracked individual in the lower-right, who interacts with the group at the end of the sequence. The mesh is pushed downward due to the influence of the dynamics of this individual, and the track fails.

Figure 5-21: Piecewise affine meshes are more robust to error and local misregistrations than are simple affine patches. Top row: the tracked affine patch loses its target in the presence of the camera flash. Bottom: the piecewise affine mesh is robust to the camera flash due to the fidelity of matches elsewhere.

Figure 5-22: Tracking groups of people is more robust than tracking individuals. The blue mesh fails to maintain the track of the individual as she is occluded by another individual within the tracked group. Instead, the mesh continues tracking the occluding figure who then leaves the group. In contrast, the red mesh continues tracking the group successfully through many occlusions and other visual distractions.

With a more articulated model, such as the one shown in Figure 5-23, the track remains robust to such distractions. In this challenging sequence, a group of people is tracked as they ascend an escalator. A multitude of the difficulties described previously occur within this sequence. A number of (untracked) people interact with the group throughout the sequence, causing numerous visual distractions. However, due to the maintenance of the shape-free appearance and the constraints on the individual mesh elements induced by shared vertices and the shape prior, we are able to maintain the track. Furthermore, the individuals within the group change their relative positions during the sequence. As was described in Section 5.5.1, this is modeled as a change in textured appearance rather than as a complex, local change in shape. Thus, we are able to continue tracking the shape and dynamics of the group as a whole.

As the group enters the escalator, they enter an area of sustained occlusion in which half of the tracked region is lost from view. In this example, we use the majority-vote scheme of occlusion reasoning described by Equation 4.28 in Section 4.6. Under this method, the pixelwise maximum likelihood (ML) estimate is computed, as in the tracking scenario described in Section 5.3. However, with the assumption that a single model must be fully responsible for describing the entire area of overlap within the imagery, we determine which model is visible via majority-vote over the visible pixels in the ML estimate. In this case, it is clear that the escalator is responsible for the majority of visible pixels within the region of overlap. As such, the occluding region is correctly inferred.

We continue to track through this large area of occlusion, and even maintain the appearance of the region within the areas of occlusion. By the end of the sequence, the leading figure in the group is entirely out of view of the camera, but we still maintain his appearance as part of the group. As was shown in Section 5.4.3, the ability to maintain and infer the appearance and shape of a region in areas where the region is not visible has applications in analysis and synthesis.

Figure 5-23: Group tracking under challenging conditions. Challenges include: interactions with untracked individuals, significant occlusion by escalator and building structure, appearance changes caused by individuals changing positions within the group, and significant changes to shape cause by a compression and articulation of the group. Top: tracked position of mesh. Bottom-left: sample of shape-free appearance. Bottom-center: appearance model. Bottom-right: estimated occlusion mask.

Figure 5-24: Lack of articulation on the part of the shape parametrization caused this track to fail. The individual in the lower right is a distraction that, due to the construction of the mesh, was unable to be ignored. The result of incorporating his dynamic into the deformation model results in a failed track.

5.6 Application: Image Denoising

When the pose of the tracked object does not change for a number of frames, such that no new visual elements are introduced or removed during that time, then the shape-free appearance of the object will be stationary. During these times, the appearance model will accurately estimate the true appearance of the object, the effect of which is to remove the noise from the imagery. Figure 5-25 shows a sample of shape-free appearance of the "red" car from the tracking run of Section 5.3, derived directly from the imagery, and the appearance model obtained from tracking the car for ten frames. The bottom set of imagery in Figure 5-25 shows an image and associated appearance model for the tracked sequence presented in Section 5.5.3. As can be seen, much of the pixelated noise that gives the imagery a grainy appearance and obscures detail is eliminated from the representation of the region maintained by the appearance model. Details, such as the text engraved on the wall and the uprights on the car, are more easily discerned within the appearance model since these finer details are not encumbered with noise. Also, the boundaries between sections become clearer, where on the car there is a clear separation between the horizontal gray detail by the tires and the red paint further up on the car.

The appearance model also mitigates the effect of interlacing. Interlacing is caused by a digital sensor only sensing every other scanline at one time. A complete image is obtained at each time t by exposing the sensor twice per frame – once for the even scanlines and once for the odd. This results in the boundaries of moving objects appearing jagged and unclear. This can be observed in the pillars of the car by the windows in the image of the car shown in Figure 5-25. These effects are mitigated in the estimate of appearance maintained by the appearance model since, after several frames of video, the appearance model will have observed many samples of even and odd scanlines, and it is unlikely that the car would maintain its precise position on those scanlines in the sensor for the duration of the sequence.

Though we do not consider it here, the shape-free measurements may themselves be used in the process of image enhancement. By assuming that the imagery arise from an "ideal" image subject to a series of transforms involving blur, downsample, and some form of planar warp, the images can be used reconstruct an estimate of the latent image from which these lower quality images are derived. Previous works on the subject assume affine [3] or projective [12] distortion as part of their models of the image formation process. Such models are used to register a sequence of images. Once registered, they can then be used to infer a higher resolution, idealized version of the object or region. In this regard, the process of tracking and the method of acquiring shape-free appearance measurements suggested here can be used within the above frameworks to allow those methods to be used on imagery undergoing more complex deformations than simple planar transformations.

5.7 Automatic Initialization of Tracks

Using the model and method of detection described in the previous chapter, we now describe a method for automatically initializing the tracks. Suppose we have a mesh such as the one shown in Figure 5-26. It is a generic mesh, not precisely fitting any object we wish to track. Furthermore, assume that there are m objects within that mesh that move distinctly, such that they are separable in terms of their joint texture and motion properties.

The goal, then, is twofold: to determine which locations, or pixels, within the mesh

are associated with which models of motion, and to determine those models of motion. Of course, if we knew where each m_i was at the current time $t = 0$, shown in Figure 5-26 and we knew where each m_i was at the next time step, we could estimate the deformation of the regions and determine occlusions masks by the process described in Section 4.6. And, if we knew the occlusion masks, computing the deformations is a straightforward task of detection described in Section 4.4.

Our solution is to iteratively update both in a coordinate-ascent framework. Given an initial guess at the occlusion masks, we find each m_i in the imagery at $t = 1$ according to Equation 4.21. This gives us an estimate of the deformations each region undergoes. Based on these detections, the shape-free appearance can then be compared via Equation 4.26 to obtain a more refined estimate of the occlusion masks. These two steps are iterated over until the estimates of deformation and occlusion masks have converged.

We validate the initialization procedure described above by segmenting a region containing two opposing motion fields. Figure 5-26 shows a generic mesh situated on top of the two cars tracked previously in Section 5.3. Using a two-frame iterative estimation process, we segment the region under the mesh into the two most likely motion fields. Figure 5-27 shows the resulting segmentation and occlusion masks estimated under a number of different starting conditions. The results indicate that the estimated masks are stable with respect to their starting condition and quickly converge to a local optima.

The noise present in the resulting segmentations is easily interpreted. Since the black car does not move as much as the red car, most of the stationary background pixels get associated with it, including those pixels associated with the street light occluding the red car. The background pixels that get associated with the red car are due to the fact that most of the background in that area is not textured, so those areas look equally likely under both motion models. This is evidenced by the random, noisy nature of the segmentations in those areas. We note that the areas in which the cars are situated have been segmented with a high degree of accuracy.

5.8 Stability

Two parameters that have a significant effect on the shape and ability to track successfully are the degree of articulation of the mesh and the strength of the shape prior, encoded by the parameter λ in the cost function of Equation 4.18, during the process of detection.

The deformation of the shape of the tracked region is modeled as a piecewise-affine mesh. Each triangular element of the mesh represents a locally affine transformation. If the mesh were composed of a single element, the deformation of the shape would be affine. As the number of mesh elements increases, the shape is allowed to deform in more complex and articulated ways. Taken to its extreme, each individual pixel within the mesh can be represented by its own mesh element, yielding a deformation of shape that resembles optical flow.

The shape prior used is a spring model that applies constraints on how much the shape is allowed to deform from one frame to the next. The spring model aims to preserve the lengths of each edge of the mesh, but not necessarily their position or orientation. In this regard, the shape prior regularizes the shape deformations of the mesh by preserving rigidity. The influence of the shape prior during the detection process is encoded by the parameter λ, which balances the effect of the shape prior and the data fidelity term. With $\lambda = 0$, the shape prior has no effect and the detection process will optimize for the best match of model to imagery. As λ increases, the shape changes become more and more constrained and the deformations increasingly resemble rigid transformation.

In this way, the strength of the shape prior and the degree of articulation have opposite effects on the ability of the shape to deform. In order to study the effects of these parameters, we ran a set of experiments on the group tracking data of Section 5.5.1. We initialize a rectangular area around a group of people and track that group as they turn the corner. A successful track is shown in Figure 5-28. To determine the effects of the two parameters, we tracked the region several times while systematically altering the degree of articulation and the strength of the shape prior between each run. The final frame of each run is shown in Figure 5-29. The degree of articulation of the shape began with a two-element mesh and was increased by the addition of interior points, laid out in a checkerboard pattern with

increasingly dense spacing. The shape prior began with $\lambda = 0$ and was increased in value by 0.0005 each time.

Seven increasingly finer articulated meshes were used and six different values of λ. The final frame of each run of the tracker with the cross product of these settings are shown in Figure 5-29. As described above, as the degree of articulation is increased, the deformation of the mesh likewise increases. The converse is true for the strength of the shape prior. However, the degree to which the shape prior influences the overall shape of the mesh is not uniform. For example, it first tends to regularize out the more complex deformations associated with the mesh deforming to match changing conditions within the shape. The strength of the shape prior needs to be significantly stronger in order to regularize out the deformations associated with a change in outline of the shape. Thus, there is a range of values, indicated in green, that produce articulated meshes that deform to follow the curve of the region as it takes the corner, without inducing a change in topology of the mesh itself – as indicated by inverted mesh elements. The tracking runs to the upper-right of the "green" runs are similar and indicate a lack of articulation. These are caused by a combination of having a shape with two few mesh elements and by the increasing strength of the shape prior. The runs in the lower-left of the "green" runs are too articulated, with inverted mesh elements. These are caused by a combination of too many mesh elements and too weak of a shape prior.

The runs in green indicate parameter combinations that can well model the deformations of this type of phenomena by maintaining the shape of the group as a whole without overfitting the shape. The results indicate that there are a broad range of settings that can produce compelling results on this type of far-field crowd data.

5.9 Time Complexity

The tracking algorithm so described has the following inputs for each tracked region: n, the number of mesh nodes, e, the number of mesh elements, and a, the number of shape-free appearance locations, or pixels. The tracker maintains m models at a time. For computing time complexity, we assume that each m_i has the maximal number of nodes, mesh

elements, and appearance coefficients, to compute upper-bounds on performance. When convenient, we make the following assumptions: $n < a$, $e < a$, $n = O(a)$, $e = O(a)$, and $e = O(n)$. We also assume that the tracked regions remain at a constant scale with respect to the size a of the shape free context.

Each stage of the algorithm has a time complexity that depends on the sizes of those input sets: predict, detect, occlusion reasoning, and update. The total time complexity for the tracker will therefore be $O(predict) + O(detect) + O(occlusion) + O(update)$, and we treat each in turn.

Prediction involves running the dynamics of the system forward in time. This requires the computation of shape state and covariance as well as the computation of appearance state and covariance. These require matrix multiplications and in general, have time complexity $O(n^2 + a^2)$, or $O(O(a)^2 + a^2)$ which is simply $O(a^2)$. However, for the dynamics used to generate the results of this chapter and with the assumptions of independence resulting in diagonal covariances, the operations are linear in the size of the inputs. The effective time complexity is therefore $O(a)$.

Detection involves iteratively solving for δp, k times. Each computation of δp involves warping the imagery and the gradient of the imagery under target p' to the shape-free context, computing the F $\frac{\partial d_f}{\partial p}$ shape Jacobians, several multiplications, and one $n \times n$ matrix inversion. Computing the warp involves computing which mesh element each pixel belongs to and warping it according to its own affine warp. For each pixel, finding the mesh element is $O(e)$ and warping it is $O(1)$. Thus, the warp is $O(ae)$. Many multiplications take place on a per-pixel or per-shape prior basis. Assuming $F = O(n)$, we have that the arithmetic on a per-pixel basis is $O(n^3)$, or $O(an^3)$ for the region. Inverting a $n \times n$ matrix is $O(n^2)$. Thus, the total time complexity for detection is $O(ae + an^3 + n^2)$, or $O(an^3)$. However, in practice it is much smaller. Because the shape priors used are over edges, only 4 elements of p are involved in $\frac{\partial d_f}{\partial p}$. Similarly, the warp jacobian for a piecewise-affine warp has a constant number of nonzero elements. Together, these reduce the number of operations on a per-pixel basis to $O(n)$, and $O(an)$ for the region. This reduces the total time complexity for detection to $O(an + n^2)$. Taking into consideration that there are methods for computing δp that do not require the costly matrix inversion [47], and the time

complexity is reduced even further to $O(an)$. This is applied k times but, since k is not dependent on the size of the input, the time complexity remains unchanged.

Occlusion reasoning involves warping the imagery onto the shape free contexts and computing likelihoods on a per-pixel basis. The warp is $O(an)$ and computing the likelihood for all pixels is $O(a)$. Comparing likelihoods is $O(am)$ and so the total time complexity is $O(an) + O(am)$. However, since the warp is already computed, we have that occlusion reasoning is $O(am)$.

Update involves warping the imagery to the shape free context and updating shape and appearance parameters. Since the warp is already computed, it adds nothing to this step. In general, update is also $O(n^2 + a^2)$, involving similar computations as the predict step. And, again, with simplifying assumptions about the independence of the uncertainties, and with the measurement matrices $\mathbf{H}_a$ and $\mathbf{H}_p$ used in practice, these operations have time complexity $O(a)$.

Thus, the total time complexity of the system is $O(m(a+an+a+a))$, or $O(anm)$. The time complexity scales linearly with the number of tracked objects, and with the size of the shape and appearance spaces. Doubling the number of tracked objects doubles the number of computations. Likewise, doubling the size of the shape-free appearance contexts or the number of mesh nodes will double the processing requirements. Since the approach scales linearly, however, its speed will likewise be affected by continual increases in processing power.

5.10 Assessment

One of the most common methods of assessing the accuracy and performance of a tracker is through direct demonstration of the tracker's ability to operate under diverse and challenging conditions (e.g. [31, 50, 49, 53, 32, 35, 5]). Under this methodology, direct inspection of the resultant tracks is compared against an implied ground truth. Such a methodology is employed here to determine the accuracy of tracking many of the examples shown previously. A variant of this is to make explicit some information about the expected behavior of the resultant tracks, for example by counting the number of tracking "mistakes" exhibited

over the course of several tracked sequences [49].

Another method of assessing the accuracy of a track is to measure the residual between observation and model [29, 31, 15]. Such methods are employed as indicators of tracking and model fidelity by measuring the difference between each detected region and the model at each time t. In general, however, this is an incorrect assumption. For instance, consider the example of the affine patch tracker shown in Figure 5-21, an example of a tracking failure. When the tracker is first confused by the camera flash, the residuals will be high. However, once the track "latches on" to its position on the wall, it will begin to model it accurately and the sum of the residuals will become very small. The indicator, in this case, is indicating how well the model is representing the region it is currently tracking. It is not an indicator of whether the track is ultimately correct, though.

When ground truth is available, for example by hand labelling points as in [36, 19] or by expert annotation as in Section 5.4.3, then the performance of the tracker can be more quantitatively described. The results can then be used as a benchmark, to compare with various other tracking methods. Ultimately, we feel that this is the best approach to take with regard to being able to comparatively evaluate trackers and is the subject of future investigation.

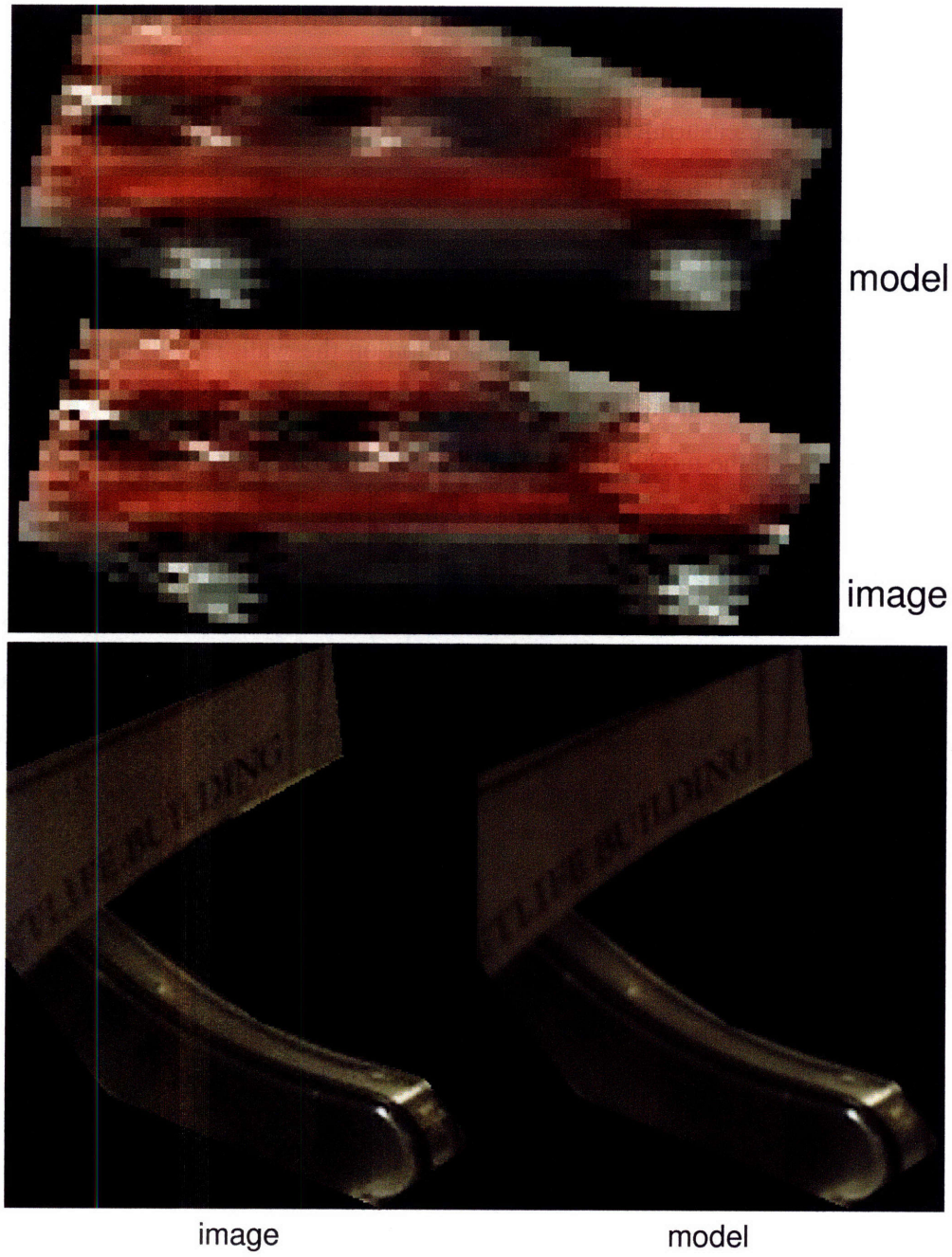


Figure 5-25: Tracking objects whose appearance is stationary for some time allows the model to accurately estimate the true appearance of the objects. The top figure shows a sample of shape-free appearance derived from the tracking sequence presented in Section 5.3 and the estimated shape-free appearance after tracking for ten frames. The bottom figure shows a sample from the tracking sequence presented in Section 5.5.3. In both cases it can be seen that the estimated appearance results in a denoised version of the imagery.

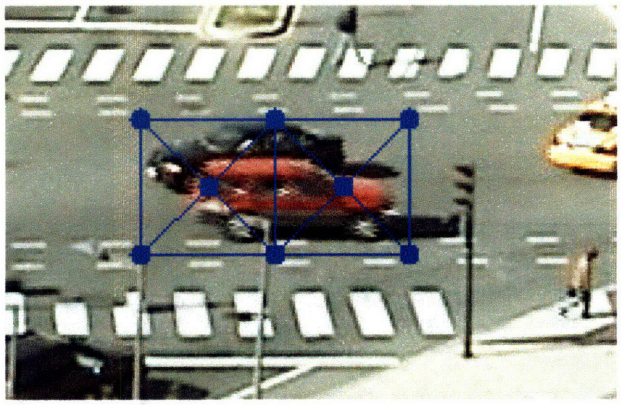


Figure 5-26: Mesh instantiated over region involving two opposing motion fields. The car in the foreground is travelling to the right and the car in the background is travelling to the left.

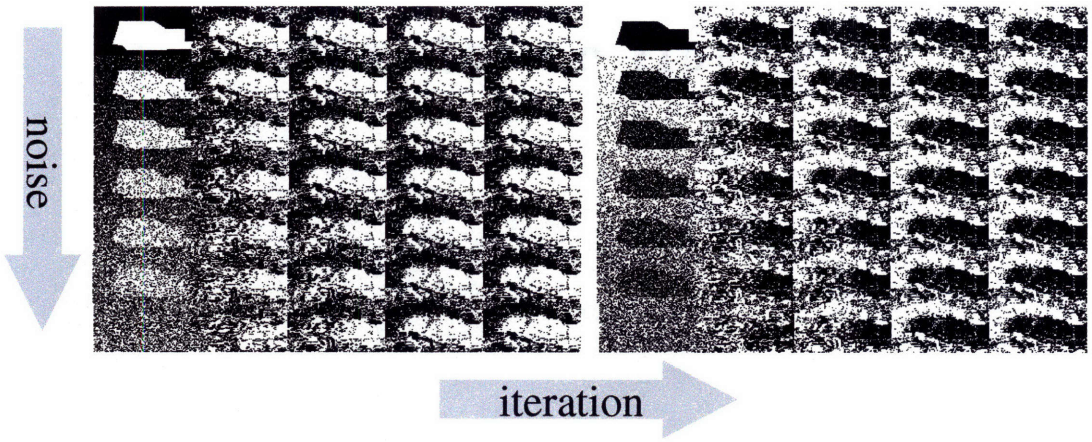


Figure 5-27: Results of automatic initialization of the model shown in Figure 5-26 under different initial conditions. The rows indicate individual runs of the initialization process with increasing amounts of noise added to the initial segmentation. The bottom row is based purely on random initialization.

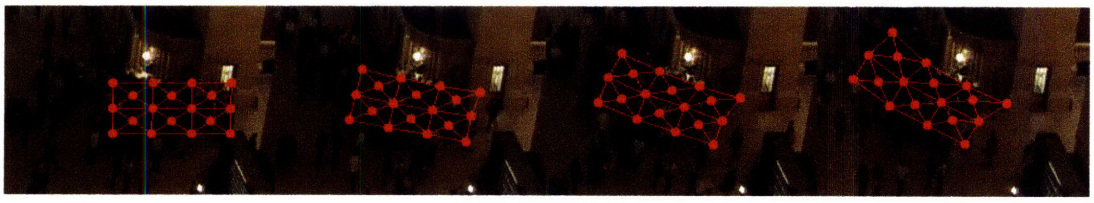


Figure 5-28: Successful tracking of a group of people as they turn a corner. The mesh is able to deform to follow the group as they turn the corner while maintaining the integrity and topology of the mesh interior.

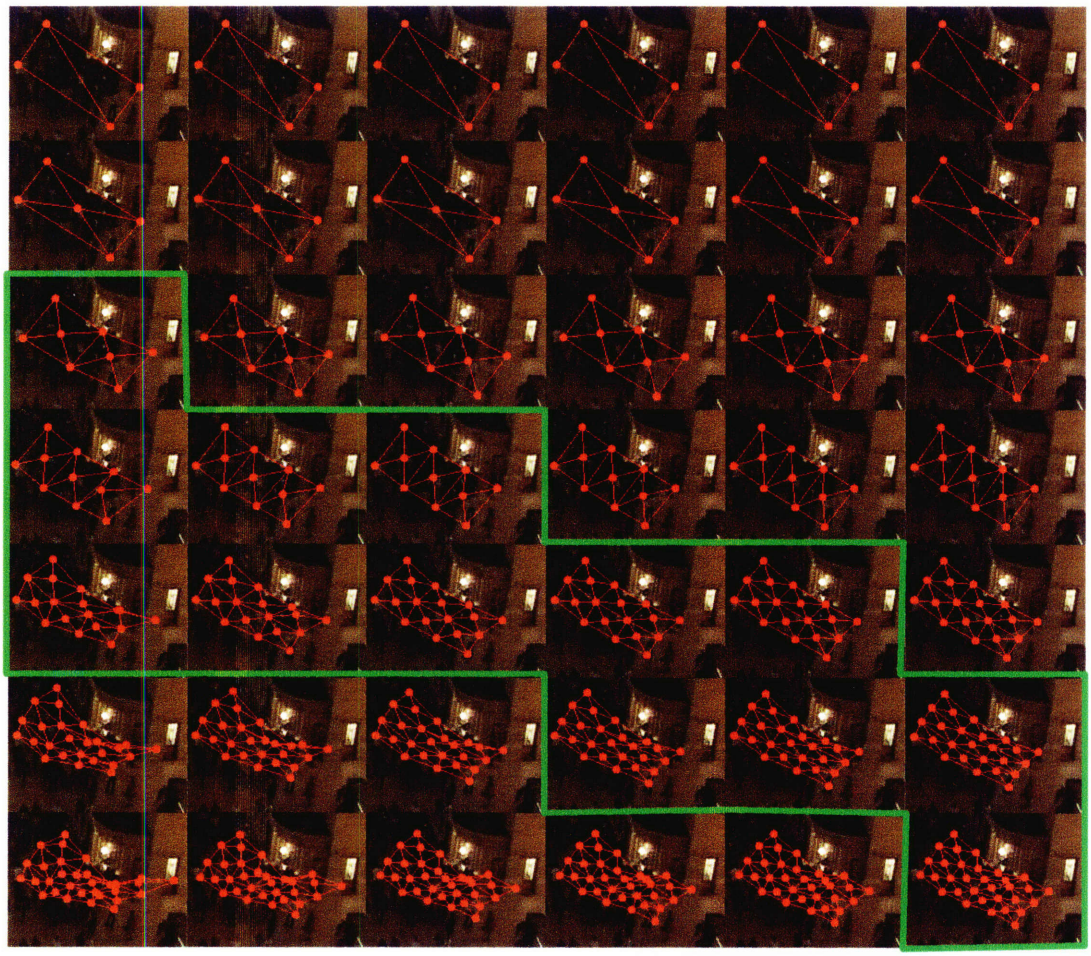


Figure 5-29: Stability as a function of articulation and rigidity. The same region was tracked using varying degrees of articulation and shape rigidity. The last frame of each run is shown above. Proceeding down adds articulation, proceeding to the right increases the rigidity of the shape, as indicated by increasing values of the shape prior weighting term λ. The runs outlined above indicate runs with sufficient articulation to model the deformation of the group as it turns the corner without inducing topological changes within the mesh. Runs to the lower-left indicate too much articulation and those to the upper-right too little.

Chapter 6

Conclusion

In this dissertation, we have introduced a model of appearance capable of representing regions of imagery that undergo complex changes to both their shape and textured appearance. Unlike similar models, the one suggested here is capable of representing visual phenomena that undergo nonrepeating changes to texture and shape and are ideally suited for online estimation and tracking.

We described how the model is used within the framework of model-based visual tracking, and described methods for predicting appearance, detecting instances of the model within the imagery, and of adapting the model to allow for the changes in appearance and shape the tracked regions can undergo. We described a method of occlusion reasoning that is both effective and computationally efficient that allows for tracked objects to interact without adversely affecting tracking performance or model accuracy. Incorporating occlusion reasoning into the tracking framework also allows for missing data to be handled in a robust and effective manner, allowing the regions to continue to be tracked even when much of it is outside the bounds of available imagery.

We showed how such models and methods of tracking enable the tracking of large groups of people in areas where traditional people tracking would be impossible. By tracking groups of people as a single, dynamic model of shape and appearance, we are able to track in densely populated scenes with such distracting elements as individuals joining, leaving, and cutting through the group. By enabling the tracking of groups, we maintain the ability to use tracking data for many applications where such coarser-grained informa-

tion is applicable, such as for source/sink analysis and for such scene modeling tasks as determining trajectories, object velocities, and for detecting unusual activity.

We showed how to use the methods of detection and occlusion reasoning to segment regions of imagery that are separable in terms of their joint texture and shape properties. By iteratively solving for the occlusion masks and deformation parameters until convergence, we arrive at segmentations in which like-labeled areas deform according to a consistent dynamic. These areas can be used to initialize the models and track objects without the need for explicit initialization.

6.1 Future Work

Based on the work described previously, we describe some avenues of future investigation.

6.1.1 Estimating Uncertainty

For the results presented in Chapter 5, we manually set the levels of uncertainty in the measurement and dynamical processes before tracking. They are set according to an a-priori estimate of the relative merit of the approximations present in both. For example, for the hurricane sequence of Section 5.1, we know that a constant velocity assumption is at best a first-order approximation to the actual dynamic governing the hurricane, whereas in the car sequence presented in Section 5.3, a constant velocity assumption is more appropriate. The noise covariances were set accordingly. This allows for efficient computation, as the gain terms can be computed offline and the state covariance remains steady. It does not, however, allow the tracker to change its prediction and update strategies based on changing conditions within the scene.

One avenue of future work is to automatically set these parameters based on a measure of variability present in the tracked data. For example, analyzing the residual $a - \mathbf{H}_a\alpha$ reveals how well the predicted shape-free appearance matches observation, and can be used to reestimate the process covariance. If the prediction abruptly becomes inaccurate, this indicates that the state has converged on an estimate that is incorrect, and so the uncertainty in that estimate and the system that produced it should increase accordingly.

6.1.2 Relation to Background Modeling

The process of tracking and the output of the detector produce an observation of shape and shape-free appearance for every frame of video, or time t. When viewed holistically, we see that the visual phenomena within shape-free observations of appearance are actually quite stationary. This is unsurprising since, during the process of tracking, shape considerations, including absolute position, orientation, and scale, are factored out and estimated within the shape model, leaving the appearance model largely unaffected by such transformations. Hence, it is a "shape-free" context.

Thus, what we are left modeling within the shape-free context of appearance is a phenomena that changes appearance, but remains stationary, as if viewed by a stationary camera. We note that this is the working assumption common to the field of background modeling. Background modeling is primarily concerned with modeling the unmoving portions of a scene imaged by a stationary camera. Though the phenomena to be modeled are unmoving in space, they are not unchanging. Tree branches rustle in the wind, construction flashers signal on and off, water ripples, and flags billow. Such complex and random motions are often handled by stationary background models.

The model of shape-free appearance we maintain assumes each pixel can be modeled as a single gaussian. This leads to the blending effect caused by complex, local deformations, such as when the positions of two people within a group change places. Similar events happen frequently within a stationary background as well. For example, a car may leave its parking spot, revealing the pavement below, or a box may be put down on top of a tiled floor. These situations were initialled handled similarly by the first background models [72, 56, 41]. However, more sophisticated approaches to background modeling handle these by maintaining several background models at a time within a mixture of Gaussians [63], or by estimating the appearance nonparametrically [21]. Such approaches could be incorporated to model the shape-free appearance of the tracked regions.

6.1.3 Initialization

We described one method of automatically initializing overlapping regions by clustering the imagery into regions whose estimated shape deformations are consistent using an iterative framework. Such a method still requires knowledge about the approximate size of the regions and, like similar methods, requires knowledge about the number of clusters to learn.

Other methods of automatically initializing regions may be more appropriate in certain circumstances. For example, Toyon, et. al suggest in [67] to use the silhouettes produced by background subtraction as an initialization step for a more sophisticated, model-based tracker. Koller [41] used this method to initialize a contour tracker for a highway scene.

6.1.4 Shape Parametrization

Another open problem related to initialization is choosing how articulate to make the model. The model must be sufficiently articulate, that is, contain enough mesh elements, to model the deformations a tracked object is likely to undergo, but should not be so complex that it is slow and prone to failure. To automatically select an optimal mesh parametrization, one can begin tracking the region with several "candidate" meshes. After several frames, the likelihood of the data under each model can be evaluated, to determine how well each model is tracking the visual phenomena. This can be combined with a minimum description length (MDL) criteria to penalize overfitting in favor of simpler models. Once the optimal parametrization is chosen, that one can be used to track for the remainder of the sequence.

6.1.5 Shape Priors

The shape priors used for detection provide robustness to visual distractors and keep the detector from overfitting to the imagery. However, the spring priors will not keep the mesh from changing topology. A change in topology occurs when a node within the mesh crosses an edge. This results in malformed, degenerate mesh elements if the node falls on the edge and inversions if it crosses it. Both indicate a tracking failure.

Different shape priors can be used to eliminate this problem. For example, by incorporating into the shape prior an energy term that is a function of the determinant of the matrix composed of the coordinates of the vertices of the mesh element (see Appendix A for details), such changes in topology can be eliminated.

Appendix A

Derivation of Unique Affine Transform

Let ΔA be a triangle with vertices at $p = \{x_1, y_1\}$, $q = \{x_2, y_2\}$, and $r = \{x_3, y_3\}$. Assume p, q, r not all collinear, so that A is nondegenerate. Let ΔB be similarly defined with vertices at $P = \{X_1, Y_1\}$, $Q = \{X_2, Y_2\}$, and $R = \{X_3, Y_3\}$. If ΔA and ΔB describe an affine transformation, then they are related by the following:

$$\begin{pmatrix} X_1 & X_2 & X_3 \\ Y_1 & Y_2 & Y_3 \end{pmatrix} = \begin{pmatrix} a & b & c \\ d & e & f \end{pmatrix} \begin{pmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{pmatrix}, \tag{A.1}$$

where the parameters a, b, c, d, e, f of the affine transform are unknown. The remainder of this proof, therefore, is to show that, when ΔA and ΔB are nondegenerate, there exists a unique setting of those parameters that satisfies Equation A.1. In doing so, we show that there exists a unique affine transform that brings points in ΔA to ΔB. Let $\alpha = \begin{pmatrix} X_1 & X_2 & X_3 \\ Y_1 & Y_2 & Y_3 \end{pmatrix}$, $\beta = \begin{pmatrix} a & b & c \\ d & e & f \end{pmatrix}$ and $\gamma = \begin{pmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{pmatrix}$. Therefore,

$$\alpha = \beta\gamma, \tag{A.2}$$

and so

$$\beta = \alpha\gamma^{-1}. \tag{A.3}$$

Now we must say something about the existence of γ^{-1} and its relation to ΔA. It is well known that the area of a triangle is related to the determinant of the matrix composed of its vertices according to:

$$area(\Delta) = \frac{1}{2} abs \left(\left| \begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{bmatrix} \right| \right). \tag{A.4}$$

We note that γ is that matrix for ΔA. So, the area of $\Delta A = \frac{1}{2} abs(|\gamma|)$. It is also well known that $|\gamma| \neq 0$ iff γ^{-1} exists. Since the area of a nondegenerate triangle is nonzero, γ^{-1} must necessarily exist. Therefore, we have:

$$\begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = \frac{1}{|\gamma|} \begin{bmatrix} X_3y_1 - y_1X_2 + y_3X_2 - X_3y_2 - y_3X_1 + X_1y_2 \\ -x_1X_3 - x_3X_2 + x_2X_3 + x_3X_1 + x_1X_2 - x_2X_1 \\ y_1x_3X_2 - X_1x_3y_2 - y_3x_1X_2 + y_3x_2X_1 - y_1x_2X_3 + X_3x_1y_2 \\ Y_1y_2 - y_2Y_3 + y_1Y_3 - Y_1y_3 + Y_2y_3 - y_1Y_2 \\ x_2Y_3 + x_3Y_1 - x_1Y_3 - x_3Y_2 - x_2Y_1 + x_1Y_2 \\ y_2x_1Y_3 - Y_2x_1y_3 - y_1x_2Y_3 + y_1x_3Y_2 + Y_1x_2y_3 - y_2x_3Y_1 \end{bmatrix}, \tag{A.5}$$

where

$$|\gamma| = -x_1y_3 + x_2y_3 - x_2y_1 - x_3y_2 + x_3y_1 + x_1y_2 \tag{A.6}$$

Bibliography

[1] S. Baker and I. Matthews. Equivalence and efficiency of image alignment algorithms. *CVPR*, 2001.

[2] Simon Baker and Iain Matthews. Lucas-kanade 20 years on: A unifying framework: Part 1. Technical Report CMU-RI-TR-02-16, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, July 2002.

[3] B. Bascle, A. Blake, and A. Zisserman. Motion deblurring and super-resolution from an image sequence. *ECCV*, 1996.

[4] B. Bascle and R. Deriche. Region tracking through image sequences. *ICCV*, 1995.

[5] M. J. Black and A. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *IJCV*, 1998.

[6] A. Blake, R. Curwen, and A. Zisserman. A framework for spatio-temporal control in the tracking of visual contours. *IJCV*, 1993.

[7] A. Blake, R. Curwen, and A. Zisserman. Learning to track the visual motion of contours. *Artificial Intelligence*, 1995.

[8] F. Bookstein. Principal warps: Thin-plate splines and the decomposition of deformations. *PAMI*, 1989.

[9] B. Bose and W.E.L. Grimson. Ground plane rectification by tracking moving objects. *Proceedings of the Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS)*, October 2003.

[10] B. Bose and W.E.L. Grimson. Improving object classification in far-field video. *CVPR*, June 2004.

[11] D. Capel and A. Zisserman. Super-resolution enhancement of text image sequences. In *ICPR*, 2000.

[12] D. Capel and A. Zisserman. Super-resolution from multiple views using learnt image models. In *CVPR*, 2001.

[13] Y. Chen, T. Huang, and Y. Rui. Parametric contour tracking using unscented kalman filter. *ICIP*, 2002.

[14] R. Collins and Y. Liu. On-line selection of discriminative tracking features. In *ICCV*, 2003.

[15] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *CVPR*, 2000.

[16] T. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. In *ECCV*, 1998.

[17] Gianfranco Doretto. Modeling dynamic scenes with active appearance. *CVPR*, 2005.

[18] Gianfranco Doretto, Alessandro Chiuso, Ying Nian Wu, and Stefano Soatto. Dynamic textures. *International Journal of Computer Vision*, 2003.

[19] N. Dowson and R. Bowden. Simultaneous modeling and tracking (smat) of feature sets. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2005.

[20] G. J. Edwards, T. Cootes, and C. J. Taylor. Advances in active appearance models. In *ICCV*, 1999.

[21] Ahmed Elgammal, David Harwood, and Larry Davis. Non-parametric model for background subtraction. In *European Conference on Computer Vision*, 2000.

[22] Andrew W. Fitzgibbon. Stochastic rigidity: Image registration for nowhere-static scenes. In *ICCV*, 2001.

[23] Andrew W. Fitzgibbon. Stochastic rigidity: Image registration for nowhere-static scenes. In *ICCV*, 2001.

[24] D. Forsyth and J. Ponce. *Computer Vision: A Modern Approach.* Prentice Hall, 2002.

[25] William Freeman and Edward Adelson. The design and use of steerable filters. In *PAMI*, 1991.

[26] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1984.

[27] T.F.Cootes G.J.Edwards, C.J.Taylor. Learning to identify and track faces in image sequences. In *Int. Conf. on Face and Gesture Recognition*, 1998.

[28] W.E.L. Grimson, C. Stauffer, R. Romano, and L. Lee. Using adaptive tracking to classify and monitor activities in a site. *CVPR*, 1998.

[29] G. Hager and P. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. In *PAMI*, 1998.

[30] B. Han, D. Comaniciu, Y. Zhu, and L. Davis. Incremental density approximation and kernel-based bayesian filtering for object tracking. *CVPR*, 2004.

[31] B. Han, Y. Zhu, D. Comaniciu, and L. Davis. Kernel-based bayesian filtering for object tracking. *CVPR*, 2005.

[32] M. Isard and A. Blake. Contour tracking by stochastic propagation of conditional density. *ECCV*, 1996.

[33] Tomas Izo and W. Eric L. Grimson. Unsupervised modeling of object tracks for fast anomaly detection. In *ICIP*, September 2007.

[34] O. Javed and M. Shah. Tracking and object classification for automated surveillance. *ECCV*, 2002.

[35] Allan Jepson, David Fleet, and Thomas El-Maraghi. Robust online appearance models for visual tracking. In *CVPR*, December 2001.

[36] Allan Jepson, David Fleet, and Thomas El-Maraghi. Robust online appearance models for visual tracking. In *PAMI*, October 2003.

[37] N. Johnson and D. Hogg. Learning the distribution of object trajectories for event recognition. *Image and Vision Computing*, 1996.

[38] N. Jojic and B. Frey. Learning flexible sprites in video layers. *CVPR*, 2001.

[39] A. Kale and C. Jaynes. A joint illumination and shape model for visual tracking. In *CVPR*, 2006.

[40] M. Kass, A.P. Witkin, and D. Terzopoulos. Snakes: Active contour models. In *IJCV*, 1988.

[41] D. Koller, J. Weber, and J. Malik. Robust multiple car tracking with occlusion reasoning. In *ECCV (1)*, 1994.

[42] J. Lim and M. Yang. A direct method for modeling non-rigid motion with thin plate spline. In *CVPR*, 2005.

[43] D. Lowe. Robust model-based motion tracking through the integration of search and estimation. *IJCV*, 1992.

[44] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI*, April 1981.

[45] F. Lv, T. Zhao, and R. Nevatia. Self-calibration of a camera from video of a walking human. *ICPR*, 2002.

[46] F. Lv, T. Zhao, and R. Nevatia. Camera calibration from video of a walking human. *PAMI*, 2006.

[47] Iain Matthews and Simon Baker. Active appearance models revisited. *International Journal of Computer Vision*, 60(2):135 – 164, November 2004.

[48] Iain Matthews, Takahiro Ishikawa, and Simon Baker. The template update problem. In *Proceedings of the British Machine Vision Conference*, September 2003.

[49] H. Nguyen, M. Worring, and R. van den Boomgaard. Occlusion robust adaptive template tracking. *ICCV*, 2001.

[50] C. Olson. Maximum likelihood template matching. *CVPR*, 2000.

[51] S. Osher and J. Sethian. Fronts propagating with curvature-dependent speed: algorithms based on the hamilton-jacobi formulation. In *Journal of Computational Physics*, 1988.

[52] Nikos Paragios and Visvanathan Ramesh. A mrf-based approach for real time subway monitoring. In *IEEE Computer Vision and Pattern Recognition*, 2001.

[53] P. Pérez, C. Hue, J. Vermaak, and M. Gangnet. Color-based probabilistic tracking. In *ECCV*, 2002.

[54] N. Peterfreund. Robust tracking with spatio-velocity snakes: kalman filtering approach. *ICCV*, 1998.

[55] Robert Pless, John Larson, Scott Siebers, and Ben Westover. Evaluation of local models of dynamic backgrounds. In *Computer Vision and Pattern Recognition*, 2003.

[56] C. Ridder, O. Munkelt, and H. Kirchner. Adaptive background estimation and foreground detection using kalman filtering. *Proc. ICAM*, 1995.

[57] S. Sclaroff and J. Isidoro. Active blobs. In *ICCV*, 1998.

[58] J. Shi and C. Tomasi. Good features to track. In *CVPR*, 1994.

[59] Stefano Soatto, Gianfranco Doretto, and Ying Nian Wu. Dynamic textures. *ICCV*, 2001.

[60] C. Stauffer. Automatic hierarchical classification using time-based co-occurrences. *CVPR*, 1999.

[61] C. Stauffer. Estimating tracking sources and sinks. *Proceedings of the Second IEEE Workshop on Event Mining*, 2003.

[62] C. Stauffer. Minimally-supervised classification using multiple observation sets. *ICCV*, 2003.

[63] C. Stauffer and W.E.L. Grimson. Adaptive background mixture models for real-time tracking. *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 1999.

[64] Chris Stauffer and W. Eric L. Grimson. Learning patterns of activity using real-time tracking. In *PAMI*, 2000.

[65] G. Strang. *Introduction to applied mathematics*. Wellesley-Cambridge Press, 1986.

[66] C.J. Taylor T.F. Cootes, D. Cooper and J. Graham. Active shape models - their training and application. *Computer Vision and Image Understanding*, 1995.

[67] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers. Wallflower: Principles and practice of background maintenance. In *ICCV (1)*, 1999.

[68] M. Turk and A. Pentland. Face recognition using eigenfaces. *CVPR*, 1991.

[69] Xiaogang Wang, Kinh Tieu, and Eric Grimson. Learning semantic scene models by trajectory analysis. *ECCV*, 2006.

[70] Gary Welch and Gary Bishop. An introduction to the kalman filter. Technical Report TR 95-041, University of North Carolina at Chapel Hill, Chapel Hill, NC 27599, July 2006.

[71] J. Winn and A. Blake. Generative affine localisation and tracking. *NIPS*, 2004.

[72] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *PAMI*, 1997.

[73] Jing Zhong and Stan Sclaroff. Segmenting foreground objects from a dynamic, textured background via a robust kalman filter. In *ICCV*, 2003.