

## MODEL BASED TRACKING OF ARTICULATED OBJECTS

Kevin Michael Nickels, Ph.D.  
Department of Electrical and Computer Engineering  
University of Illinois at Urbana-Champaign, 1998  
Seth Hutchinson, Advisor

The goal of this research is to develop a computer program that will track a complex, articulated object. We assume that we know the appearance and kinematic structure of the object to be tracked, leading to what is typically termed a *model-based* object tracker. This tracker observes a monocular grayscale image of the scene and combines information gathered from this image with knowledge of the previous configuration of the object to estimate the configuration of the object at the time the picture was taken. Each degree of freedom in the model has an uncertainty associated with it, indicating the confidence in the current estimate for that degree of freedom. The uncertainty estimates are updated after each observation.

Unique aspects of this work include on-line model-based generation of complex features for use as templates, the characterization of uncertainties in point feature motion for use in assimilating feature tracking results, the use of the sum-of-squared-differences (SSD) image correlation measure as a measurement of the observation error in feature tracking, and the use of complex articulated object models in tracking.

© Copyright by Kevin Michael Nickels, 1998

MODEL BASED TRACKING OF ARTICULATED OBJECTS

BY

KEVIN MICHAEL NICKELS

B.S., Purdue University, 1993

M.S., University of Illinois, 1996

THESIS

Submitted in partial fulfillment of the requirements  
for the degree of Doctor of Philosophy in Electrical Engineering  
in the Graduate College of the  
University of Illinois at Urbana-Champaign, 1998

Urbana, Illinois

## ABSTRACT

The goal of this research is to develop a computer program that will track a complex, articulated object. We assume that we know the appearance and kinematic structure of the object to be tracked, leading to what is typically termed a *model-based* object tracker. This tracker observes a monocular grayscale image of the scene and combines information gathered from this image with knowledge of the previous configuration of the object to estimate the configuration of the object at the time the picture was taken. Each degree of freedom in the model has an uncertainty associated with it, indicating the confidence in the current estimate for that degree of freedom. The uncertainty estimates are updated after each observation.

Unique aspects of this work include on-line model-based generation of complex features for use as templates, the characterization of uncertainties in point feature motion for use in assimilating feature tracking results, the use of the sum-of-squared-differences (SSD) image correlation measure as a measurement of the observation error in feature tracking, and the use of complex articulated object models in tracking.

Dedicated to my wife, Elizabeth Gertrude Griffin Furlong Nickels

## ACKNOWLEDGMENTS

I would like to thank my wife, Elizabeth, for her love and understanding while this work was going on. Due to being a sounding board for my ideas and a wonderful proofreader, she now knows more about computer vision than most other biophysical chemists.

I would like to thank my parents for bringing me up with the inquisitive and adaptive mind-set needed to conceive this research, and the self-discipline to execute it.

I would like to thank my advisor, Professor Seth Hutchinson, for providing a rich, creative environment in which to work. His ideas and guidance have been invaluable.

I would like to thank my peers in the Computer Vision and Robotics research group for their many helpful discussions. Without the many discussions in our lab group, this research would not have been nearly as interesting as it has been. I would particularly like to thank Becky Castaño, Peter Leven, and Yakup Genc for their input.

# TABLE OF CONTENTS

CHAPTER	PAGE
<b>1 INTRODUCTION</b> . . . . .	1
<b>2 LITERATURE REVIEW</b> . . . . .	5
2.1 Tracking via Image Flow . . . . .	5
2.2 Tracking via Feature Correspondence . . . . .	7
2.3 Region Tracking . . . . .	7
2.4 Feature Location Prediction . . . . .	8
2.5 Feature Search . . . . .	9
2.5.1 Simplify the search . . . . .	9
2.5.2 Line features . . . . .	10
2.5.3 Correlation and feature templates . . . . .	11
2.5.3.1 Template content . . . . .	11
2.5.3.2 Similarity metrics . . . . .	12
2.5.3.3 Feature location with similarity metrics . . . . .	17
2.5.3.4 Confidence measures . . . . .	18
2.6 Object Tracking . . . . .	23
2.7 Kalman Filtering in Feature and Object Tracking . . . . .	25
<b>3 KALMAN FILTERING</b> . . . . .	30
3.1 Basic Kalman Filtering . . . . .	31
3.1.1 System definition . . . . .	31
3.1.2 Decomposition of the system . . . . .	31
3.1.3 Solution for stochastic system . . . . .	33
3.1.3.1 No history . . . . .	34
3.1.3.2 Solution with observation history . . . . .	35
3.1.4 Recursive solution for stochastic system . . . . .	36
3.1.4.1 Observation update . . . . .	37
3.1.4.2 Time update . . . . .	39
3.1.4.3 Kalman gain and covariance updates . . . . .	40
3.1.4.4 Summary . . . . .	42
3.1.5 Combining deterministic and stochastic solutions . . . . .	42
3.2 Extended Kalman Filter . . . . .	43
3.3 Alternate Form of Extended Kalman Filter . . . . .	46
3.4 An Intuitive Look at the Kalman Filtering Equations . . . . .	49

3.5	Robustness and Limitations of Kalman Filtering . . . . .	54
<b>4</b>	<b>A MODEL-BASED OBJECT TRACKING SYSTEM . . . . .</b>	<b>56</b>
4.1	Algorithm . . . . .	56
4.2	Feature Tracking . . . . .	56
4.2.1	Models used . . . . .	59
4.2.1.1	System model . . . . .	59
4.2.1.2	Object geometric model . . . . .	60
4.2.1.3	Object appearance model . . . . .	60
4.2.1.4	Imaging model . . . . .	61
4.2.1.5	Object dynamic model . . . . .	61
4.2.1.6	Errors in models . . . . .	62
4.2.2	Scene rendering . . . . .	63
4.2.3	Generating feature appearance templates . . . . .	65
4.2.4	Use of SSD in feature tracking . . . . .	67
4.2.5	Spatial discrimination . . . . .	68
4.2.6	Approximation for $\mathcal{RD}$ . . . . .	71
4.2.6.1	Gaussian random vector . . . . .	72
4.2.6.2	Parameter estimation from the SSDS . . . . .	75
4.3	Some Comments on Feature Selection . . . . .	76
4.4	Assimilation of Feature Tracking Results . . . . .	80
4.5	Treatment of Feature Occlusion . . . . .	81
4.5.1	Features off-screen . . . . .	81
4.5.2	Self-occlusion of features . . . . .	82
4.5.3	External occlusion of features . . . . .	83
<b>5</b>	<b>APPLICATION OF FEATURE TRACKING . . . . .</b>	<b>85</b>
5.1	Point Feature: Gripper Finger . . . . .	87
5.2	Edge Feature . . . . .	88
5.3	Degenerate Point Feature . . . . .	94
5.4	Externally Occluded Point Feature . . . . .	94
<b>6</b>	<b>APPLICATION OF KALMAN FILTERING FRAMEWORK . . . . .</b>	<b>98</b>
6.1	Case 1: The Base Case . . . . .	100
6.1.1	System and filter definition . . . . .	100
6.1.2	Filter analysis . . . . .	101
6.1.2.1	Weighting of predictions and observations . . . . .	101
6.1.2.2	Uncertainty reduction . . . . .	103
6.1.2.3	Convergence of filter . . . . .	104
6.2	Case 2: Elevated 2D Camera . . . . .	106
6.2.1	System and filter definition . . . . .	107
6.2.2	Filter analysis . . . . .	109
6.2.2.1	Weighting of observations and predictions . . . . .	109
6.2.2.2	Uncertainty reduction . . . . .	111
6.3	Case 3: More Complex Models . . . . .	112
6.3.1	System and filter definition . . . . .	114
6.3.2	Filter analysis . . . . .	117



6.3.2.1	Weighting of predictions and observations . . . . .	118
6.3.2.2	Uncertainty reduction . . . . .	119
6.4	Case 4: Use of Dynamic Model . . . . .	123
6.5	Case 5: 3DOF PUMA Robotic Arm . . . . .	125
6.5.1	System and filter definition . . . . .	125
6.6	Case 6: Incomplete Object Model . . . . .	127
6.6.1	System and filter definition . . . . .	127
6.7	Extensions to Other Situations . . . . .	128
<b>7</b>	<b>RESULTS</b> . . . . .	<b>129</b>
7.1	Case 5a: Fixed Feature Set . . . . .	129
7.2	Case 5b: Self-Occlusion of Features . . . . .	129
7.3	Case 5c: External Occlusion of Features . . . . .	131
7.4	Case 5d: Three DOF PUMA Arm . . . . .	131
7.5	Case 6: Two DOF Arm With Unknown Link Lengths . . . . .	135
7.5.1	Case 6a: Correct link lengths . . . . .	135
7.5.2	Case 6b: Incorrect link lengths . . . . .	135
7.5.3	Case 6c: Incorrect link lengths, initial motion . . . . .	138
<b>8</b>	<b>CONCLUSIONS AND FUTURE WORK</b> . . . . .	<b>144</b>
8.1	Conclusions . . . . .	144
8.2	Future Work . . . . .	145
	<b>APPENDIX A CASE 1: BASE CASE</b> . . . . .	<b>147</b>
A.1	Projection Equations . . . . .	147
A.2	Partial Differentials . . . . .	148
A.3	Dynamic Model . . . . .	148
A.4	Partial Differentials . . . . .	148
	<b>APPENDIX B CASE 2: ELEVATED CAMERA</b> . . . . .	<b>149</b>
B.1	Projection Equations . . . . .	149
B.1.1	Transform derivation from end-effector coordinates to base coordinates . . . . .	149
B.1.2	Derivation of transform from base coordinates to the camera frame . . . . .	150
B.1.3	Combine and add perspective projection . . . . .	152
B.2	Partial Differentials . . . . .	153
B.3	Dynamic Model . . . . .	153
B.4	Partial Differentials . . . . .	153
	<b>APPENDIX C CASE 3: MORE COMPLEX MODELS</b> . . . . .	<b>154</b>
C.1	Projection Equations . . . . .	154
C.1.1	Transform derivation from end-effector coordinates to base coordinates . . . . .	154
C.1.2	Derivation of transform from base coordinates to the camera frame . . . . .	156
C.1.3	Combine and add perspective projection . . . . .	158
C.2	Partial Differentials . . . . .	160
C.3	Dynamic Model . . . . .	162
C.4	Partial Differentials . . . . .	162

<b>APPENDIX D CASE 4: DYNAMIC MODEL</b> . . . . .	163
D.1 Projection Equations . . . . .	163
D.1.1 Transform derivation from end-effector coordinates to base coordinates . .	163
D.1.2 Derivation of transform from base coordinates to the camera frame . . . .	164
D.1.3 Combine and add perspective projection . . . . .	166
D.2 Partial Differentials . . . . .	167
D.3 Dynamic Model . . . . .	168
D.4 Partial Differentials . . . . .	168
D.5 Definition of Tracking Filter . . . . .	169
 <b>APPENDIX E CASE 5: 3DOF PUMA ROBOTIC ARM</b> . . . . .	 171
E.1 Projection Equations . . . . .	172
E.2 Dynamic Model . . . . .	174
E.3 Partial Differentials . . . . .	174
 <b>APPENDIX F CASE 6: INCOMPLETE OBJECT MODEL</b> . . . . .	 176
F.1 Projection Equations . . . . .	177
F.2 Dynamic Model . . . . .	178
F.3 Partial Differentials . . . . .	179
 <b>REFERENCES</b> . . . . .	 180
 <b>VITA</b> . . . . .	 187

## LIST OF TABLES

<b>Table</b>		<b>Page</b>
1.1	Notational conventions. . . . .	3
3.1	Notation used in this section. . . . .	33
3.2	Notation used in this section. . . . .	44
6.1	Models used in Case 1. . . . .	100
6.2	Models used in Case 2. . . . .	106
6.3	Models used in Case 3. . . . .	112
6.4	Models used in Case 4. . . . .	123
6.5	Models used in Case 5. . . . .	125
6.6	Models used in Case 6. . . . .	127

## LIST OF FIGURES

Figure	Page
1.1 Example tracking scenario. . . . .	3
2.1 The use of object models. . . . .	24
3.1 Plant and measurement models. . . . .	32
3.2 Conditional density of bearing based on measured value $z_1$ . . . . .	50
3.3 Conditional density of bearing based on measured value $z_2$ . . . . .	51
3.4 Conditional density of bearing based on measured values $z_1$ and $z_2$ . . . . .	52
4.1 TRACK-OBJECT, an algorithm for object tracking. . . . .	57
4.2 Tracking system overview. . . . .	58
4.3 Individual link graphics objects. . . . .	64
4.4 Link coordinate frames. . . . .	66
4.5 A synthetic image. . . . .	67
4.6 Use of similarity metric. . . . .	68
4.7 An edge feature. . . . .	69
4.8 A corner feature. . . . .	70
4.9 A homogeneous feature. . . . .	70
4.10 Density function for a 2D Gaussian random vector. . . . .	72
4.11 Point feature characteristics. . . . .	73
4.12 Horizontal edge characteristics. . . . .	73
4.13 Vertical edge characteristics. . . . .	74
4.14 Diagonal edge characteristics. . . . .	74
4.15 Approximation of response distribution by density function. . . . .	76
4.16 PUMA robotic arm. . . . .	78
4.17 Reduced system model. . . . .	78
4.18 Example of off-screen features. . . . .	82
4.19 Example of self-occlusion. . . . .	83
4.20 Example of external occlusion. . . . .	84
5.1 Portion of tracking system discussed in this chapter. . . . .	86
5.2 Features tracked in this section. . . . .	88
5.3 Tracking results for gripper feature. . . . .	89
5.4 Tracking results for edge feature. . . . .	90

5.5	Tracking results for edge feature. . . . .	91
5.6	Tracking results for edge feature. . . . .	92
5.7	Tracking results for edge feature. . . . .	93
5.8	Tracking results for nondegenerate point feature. . . . .	95
5.9	Tracking results for degenerate point feature. . . . .	96
5.10	Tracking results for externally occluded point feature. . . . .	97
6.1	Portion of tracking system discussed in this chapter. . . . .	99
6.2	Case 1. . . . .	100
6.3	Observation weighting factor for Case 1. . . . .	103
6.4	$\mathbf{P}_{k,k}/\mathbf{P}_{k,k-1}$ : Uncertainty reduction from an observation. . . . .	104
6.5	Convergence properties of Case 1. . . . .	105
6.6	Case 2. . . . .	106
6.7	Camera coordinate frame. . . . .	107
6.8	Observation weighting factor matrix for Case 2, 6° elevation. . . . .	110
6.9	Observation weighting factor matrix for Case 2, 45° elevation. . . . .	111
6.10	$\mathbf{P}_{k,k}/\mathbf{P}_{k,k-1}$ : Uncertainty reduction from an observation. Case 2, 6° elevation. . . . .	112
6.11	$\mathbf{P}_{k,k}/\mathbf{P}_{k,k-1}$ : Uncertainty reduction from an observation. Case 2, 45° elevation. . . . .	113
6.12	Case 3. . . . .	113
6.13	Configuration space for Case 3. . . . .	114
6.14	Observation weighting factor matrix for 45° camera elevation. . . . .	118
6.15	Observation weighting factor matrix for 6° camera elevation. . . . .	120
6.16	Error covariance matrix for 45° camera elevation. . . . .	121
6.17	Error covariance matrix for 6° camera elevation. . . . .	122
7.1	Tracking results for Case 5a. . . . .	130
7.2	Tracking results for Case 5b. . . . .	132
7.3	Tracking results for Case 5c. . . . .	133
7.4	Tracking results for Case 5d. . . . .	134
7.5	Initial conditions for Case 6a. . . . .	135
7.6	Tracking results for Case 6a: Link lengths. . . . .	136
7.7	Tracking results for Case 6a: Joint angles. . . . .	137
7.8	Initial conditions for Case 6b. . . . .	138
7.9	Tracking results for Case 6b: Link lengths. . . . .	139
7.10	Tracking results for Case 6b: Joint angles. . . . .	140
7.11	Initial conditions for Case 6c. . . . .	141
7.12	Tracking results for Case 6c: Link lengths. . . . .	142
7.13	Tracking results for Case 6c: Joint angles. . . . .	143

# CHAPTER 1

## INTRODUCTION

The human interface to computer systems is becoming increasingly multimodal. Not long ago, the introduction of the graphical user interface and the mouse in addition to the traditional keyboard was heralded as a great step forward. But now the HCI (human-computer interface) shows promise for evolving from a keyboard and mouse to more natural forms of input, such as speech, gestures, and facial expressions.

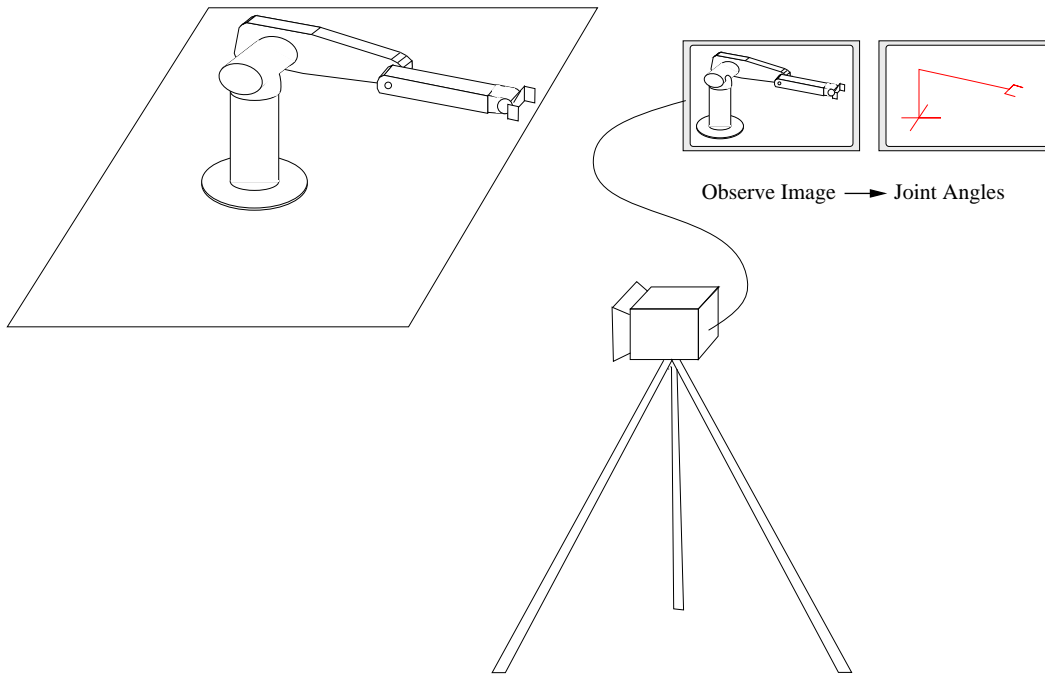
To accept gestures or facial expressions as input, the computer must first be able to *track*, or trace, the movements of the hands [1] or face [2] of the user. This requirement is one of the motivations for research in the broader computer vision area of *object tracking*. In computer vision, object tracking is the problem of following the movements of a particular object. This is a special case of the more general problems of object recognition and pose estimation: an additional constraint is placed on the problem that the object identity and pose in the previous image in a sequence is known, and the problem is to solve for them in the current image. The time between the acquisition of images in a sequence is assumed to be small enough that certain assumptions about consistency of the scene are valid, thus providing additional constraints on the problem. Object recognition is the task of identifying one of a number of previously seen objects from an image or sequence of images. Pose estimation is the problem of solving for the position and orientation of an object in an image or image sequence. Since the pose of an object affects the appearance of that object, the problem of pose estimation is often intertwined with that of object recognition.

Note that *object tracking* as we have described it above is somewhat different from the definition used by many robotics and computer vision researchers. In robotics, or visual servoing in particular, tracking is taken to mean following the movement of an object with a robotic arm, keeping a constant (or at least well-defined) transform between the end effector of the arm and the object [3], [4]. The position, and possibly orientation, of the arm are the goal, and redundancy in the specification of the configuration of the arm is acceptable. Tracking is sometimes taken to mean following the centroid or outline of one or many objects [5], [6], [7]. Here, we take the more restrictive definition of tracking, the updating of the pose and internal parameters of the object model.

A typical object tracking scenario is shown in Figure 1.1. The camera is observing the moving robot. From the image of the robot alone, the problem is to estimate all the joint parameters of the robot. This is a useful experimental setup because we have absolute joint angle information for the robot available for comparison with the joint angle estimates generated by the system, thus facilitating quantitative evaluation of system performance.

We assume that the geometric relationships among the camera, object, and base coordinate frames are known, and only the internal degrees of freedom of the object are to be estimated. We additionally assume that the initial configuration of the object is known. Often the type of motion allowed in tracking is constrained [8], [9]. Sometimes this constraint is on the relative motion of the camera and the object, as in our case [3], and sometimes the constraint is on the internal motion of the object, for example by assuming rigid objects [10].

We assume that the shape and appearance of the object being tracked are known. This assumption, also not an uncommon one [11], allows us to generate *templates* for the predicted appearance of features of interest in a given configuration. An alternative to this assumption is to use features gathered on-line [12], which may help to ensure the quality of the features tracked, since feature templates exactly match previous feature appearance. However, this method destroys any *a priori* information about the geometric relationship of the individual features to the object.



**Figure 1.1** Example tracking scenario.

**Table 1.1** Notational conventions.

$\mathbf{P}_k, \mathbf{R}_k$	Capital boldface letters	matrices
$\mathbf{z}_k, \mathbf{x}_k$	Lowercase boldface letters	vectors
$u_p, v_p$	Lowercase letters	scalars
$\hat{\mathbf{x}}_k, \hat{\mathbf{z}}_k$	symbols with circumflex above them	estimates of quantity under circumflex

Throughout this thesis, a consistent notational convention will be used. This convention is given in Table 1.1. Notation specific to the system under study will be presented in the appropriate section.

There is a large body of work on object tracking. In Chapter 2, we review the most relevant portions of the literature.

In Chapter 3, we review the derivation of the Kalman filter (KF), and its nonlinear counterpart, the extended Kalman filter (EKF). We also present an algebraically equivalent filter more suited to our particular tracking scenario.



In Chapter 4, we describe our tracking system. This system updates the internal parameters of an object model from monocular grayscale images of that object. The system accounts for both self-occlusion and external occlusion of features and weights feature observations according to their expected value in disambiguating the state, as well as the amount of spatial uncertainty present in the feature observation.

In Chapter 5, we illustrate the theory for feature tracking developed in Chapter 4 by presenting results for several features. These results will be used by the system to estimate object motion.

In Chapter 6, we develop equations for the theory described in Chapter 4 for several increasingly complex tracking situations. Each situation illustrates a particular portion of the overall tracking system.

In Chapter 7, we illustrate the effectiveness of the tracking system in several situations. Again, each situation illustrates a particular feature of the tracking system.

Finally, in Chapter 8, we offer some conclusions that can be drawn from this research, and investigate possible future work that can be done with the framework presented in this thesis.

## CHAPTER 2

### LITERATURE REVIEW

There are two main approaches to object tracking [13]. The first derives an *optical flow field*, or a dense motion field, for the sequence, then analyzes the structure of the flow field to infer structure, motion, or both for the objects in the image. This approach to object tracking is described in Section 2.1. The second approach is based on the correspondence of discrete features on an object in one image with those features in a subsequent image. This approach typically searches for the locations of features in an image by template matching or through other search techniques, and then infers object motion from these correspondences. We will discuss this approach in three separate steps. The prediction of the location of features in the image plane is discussed in Section 2.4, the search in an image for a feature is discussed in Section 2.5, and the use of the results from this search for the purpose of object tracking is reviewed in Sections 2.6 and 2.7.

#### 2.1 Tracking via Image Flow

*Image flow*, or optical flow, assigns to each point in the image plane a two-dimensional (2D) velocity vector that is the projection of the apparent three-dimensional (3D) velocity of a scene point onto the image plane [14]. This flow field is then analyzed to recover information about the dynamic scene and, in the case of object tracking, to make inferences about the motion of 3D objects in the scene.

It is well known that true velocity can be recovered with local measurements only in areas with sufficient local intensity variation [15]. Thus, computation of image flow usually occurs in two steps: using the image intensity distribution in small neighborhoods with sufficient variation to compute local information about velocity, then propagating this information into neighboring areas of the image to recover the full image flow.

There are three major approaches to recovering image flow from local information: correlation, intensity gradients, and spatiotemporal energy. Liu et al. [16] compared the performance of many of the following algorithms. Correlation searches the area surrounding each pixel in an image for a pixel in the next image with similar local neighborhood structure. The pixel in the new image with the most similar local neighborhood structure is taken to be the new location of the pixel. A typical example of the use of correlation in computing image flow can be found in Singh [17]. Gradient-based methods compute velocity from first-order derivatives of image intensity, or from filtered versions of the image. Horn and Schunk [18] are usually credited with pioneering this approach. The spatiotemporal energy approach makes some assumptions about the smoothness of the flow field in space and time, and uses 3D (space-time) Gabor filters to estimate the power spectrum of a moving pattern. Adelson and Bergen [19] presented several spatiotemporal energy models for the perception of image flow. Heeger [20] introduced the use of spatiotemporal filters in computing image flow. Fleet and Jepson [21] used the phase-space response of similar filters to compute optic flow. Weber and Malik [22] refined this approach, and gave a systematic analysis of the error sources in this approach.

There are two standard constraints used in the propagation of initial velocity field estimates to areas of insufficient local texture: smoothness of motion field and the analysis of interframe structure the motion field. The smoothness of motion field constraint assumes that the flow field is a continuous vector field [23]. An approach that uses the structure of the flow field might, for example, segment the optic flow map according to object identities and propagate velocity estimates anisotropically based on this segmentation map. A review of this research can be found in [9].

After the optical flow field is computed, the structure of the field can be analyzed to arrive at tracking results. Adiv [24] analyzed flow fields to arrive at 3D motion estimates for several moving objects.

## 2.2 Tracking via Feature Correspondence

An alternative to computing the motion of each pixel in each frame of a sequence is to only find correspondences between significant points in successive frames [25]. These significant points are called *features*. A *feature* can be any easily observable characteristic of the object being tracked. Commonly, edges, corners, and areas of high visual contrast are used as features in tracking research [10], [26], [27], although any measurable relationship in an image could be used [28].

There are two main advantages to correspondence-based feature tracking over the image-flow approach described above. Since the velocity of only a few points in the image is measured, the computational burden is reduced significantly. Also, the ability is gained to choose salient features such that the probability of proper feature velocity measurements is increased. This can be done by assessing the local image structure and choosing feature points with strong invariant properties. An invariant property is any property that does not vary with respect to changes in nuisance parameters. A nuisance parameter is any parameter not of interest in the current situation. Shi and Tomasi [29] and Papanikolopoulos [12] independently proposed confidence measures to use for the purpose of feature selection.

## 2.3 Region Tracking

A common special case of feature tracking as described above occurs when the features to be tracked are regions in an image. A *region* is often defined as a maximal homogeneous image patch, by some definition of homogeneity. Bregler and Malik [30] used region tracking to analyze the movements of humans in image sequences, for the purpose of recovering the joint

angles of the humans. The tracking of a region is analogous in many ways to the tracking of the contour surrounding the region. Contour tracking is described below.

## 2.4 Feature Location Prediction

Since features are often assumed to be spatiotemporally dense, keeping track of a feature's position and velocity over time can lead to better prediction of a feature location in the next image in a sequence. Many models can be assumed for the dynamic behavior of a feature, with the standard tradeoff between predictive power and complexity.

We will denote the position of a feature in an image taken at time  $k$  to be  $\mathbf{z}_k$ . The predicted position of a feature at time  $k$  is denoted  $\hat{\mathbf{z}}_k$ . Both  $\mathbf{z}_k$  and  $\hat{\mathbf{z}}_k$  are 2D vectors in our case.

The simplest case is to impose no dynamic model, and rely entirely on the search step to locate the appropriate features in the image. In this case, the predicted location of a feature in the next image is the same as the tracked location in the last image, or

$$\hat{\mathbf{z}}_{k+1} = \mathbf{z}_k.$$

If the motion of a feature is unpredictable, or if the interframe movement of a feature is expected to be very small, this is often a reasonable solution to the prediction problem. Singh and Allen [14] for example use a constant position dynamic model in their computation of image flow.

If the motion of a feature is predictable, a more complicated dynamic model can be beneficial. For example, the velocity of a feature in the image plane  $\mathbf{v}_k$  can be estimated and used to help predict the location of the feature in the next image. If the assumption is made that features move in the image plane with constant velocity, then

$$\hat{\mathbf{z}}_{k+1} = \mathbf{z}_k + \Delta \mathbf{v}_k,$$

where  $\Delta$  is the time between subsequent images. The constant velocity motion model is used by Clark and Zisserman [31] to assess collision probabilities for independent moving objects, and by Wilson [3] to estimate the relative position and orientation of a rigid object. This

*particle motion* concept can be carried further, estimating the acceleration of a feature in the image plane  $\mathbf{a}_k$  as well as the velocity. Then a constant acceleration model for motion could be assumed, taking

$$\hat{\mathbf{z}}_{k+1} = \mathbf{z}_k + \Delta \mathbf{v}_k + \Delta^2 \mathbf{a}_k.$$

Deriche and Faugeras [32] use a constant acceleration dynamic model for tracking line segments in an image sequence.

The dynamic model can be used to predict motion in 3D as well as motion in the image plane. Bradshaw et al. [33] use constant speed, constant velocity, and constant direction 3D dynamic models to estimate the motion of objects. A good overview of the use of dynamic models in object tracking can also be found in [33].

If the relationship between features is known, for example as part of an active contour object model [34] or part of a 3D object model [26], the configuration and motion of the encompassing object model can be used as a basis for feature location prediction. Section 2.6 describes the use of feature tracking results to update object models in this fashion.

## 2.5 Feature Search

Whatever model is used for the prediction of feature locations in an image, once locations for each feature in the image have been predicted, there needs to be some process for locating the features in the image. There are a variety of techniques used for this location, ranging from very simple techniques such as mounting LEDs on the vertices of a polyhedral object to very complex search techniques requiring specialized parallel hardware. We refer to this process as *feature search* because the image plane is being searched for the features of interest.

### 2.5.1 Simplify the search

The simplest solution to the search problem, often taken by researchers whose primary interest lies in other areas, is to simplify the corresponding computer vision problem. Rizzi

and Koditschek used first moments of a blob to track a white ping-pong ball against a black backdrop [35]. Ishii et al. mounted LEDs on a manipulator likewise to reduce the problem of search to a simple one [36].

### 2.5.2 Line features

Lines and edges are popular choices for features to track, due to their robustness to illumination variance and ease of extraction from the image.

Stephens [37] presents a local Hough-like transform to find edges in an image, and compares this to the local transform of his object model to find the image region with the most similar line structure.

Line extraction is less sensitive to noise than point extraction, and line correspondence is usually an easier problem to solve than point correspondence. Therefore, many researchers use edge-detected images as input, and the search problem becomes one of matching line segments in an image to line segments in the next image.

Bray [38] tracks edgelets using a flow algorithm, then uses the results to update a rigid polyhedral object model in 3D. *Edgelets* are short line segments in input images. Unlike many tracking research, Bray presented an initialization method for the tracking, using a model-based search based on edge matching.

Deriche and Faugeras [32] compared two representations for line segments and performed an uncertainty analysis on the parameters of each. They used a Kalman-filtering-based scheme to track line segments with each representation.

Gennery [10] uses edge-detected images in a modified Kalman filter framework to track known 3D rigid objects. At each step, a weighted least-squares method is used to match edges in the object model to edgelets in the input images.

Lowé [26] uses least-squares techniques to fit a model line segment to many smaller edge segments in the image. The overconstrained nature of the problem (there are many more edge-

segments from the object in an image than are necessary to determine the pose of an object) makes this type of search robust to noise.

Kosaka and Kak [39] use a one-dimensional (1D) search in the area predicted by their environmental model to track lines corresponding to the intersection of floors and walls, or doorways, in a hallway.

### 2.5.3 Correlation and feature templates

A feature template may also be used to detect a feature in an image. A feature template contains some representation of the feature and is compared against portions of an image to locate that feature in the image. In this case, a similarity metric is used to rate the similarity of the template and the image patch. The image region found to be the most similar to the template is usually taken to be the location of the feature.

The following sections discuss previous approaches taken in three areas crucial to correlation based tracking: the content of the template, the definition and use of a similarity metric for tracking, and the definition of confidence measures on the tracking results.

#### 2.5.3.1 Template content

In this situation, the content of the template is an important choice. A straightforward solution is to have some canonical view of the feature and to do template matching in a search window centered about the predicted position of the image. Brunelli and Poggio give a good review of this technique in the context of facial feature tracking [40]. The main problem with this approach is that the simple template is a 2D entity, and the image patch may undergo transformations that the template cannot model, such as rotation.

A more complex algorithm that also works in certain situations is to use an image patch from the previous image, taken from the area around the last computed position of the feature in that image, for the template. Hager [27] uses this approach for visual servoing. The main difficulty with this approach is feature skew. Feature skew occurs when the computed position



of the feature is slightly incorrect. This causes the next image region to be slightly offset and the next computed position of the feature is more incorrect. Slowly, the computed position migrates off the feature of interest, and the template contains image structure adjacent to the feature of interest.

If object and scene modeling are part of the tracking framework, it is possible to create templates from this information. Li et al. [41] use the tracked parameters of the CANDIDE head model to compute the expected appearance of facial features. Tang [42] and Lopez et al. [11] have a 3D registered texture of a face as part of their object model. Computer graphics techniques are used to render the relevant portion of the scene complete with sophisticated texture mapping to estimate the appearance of a feature in the image. This image patch is then used as a template in the feature tracking portion of the system.

### 2.5.3.2 Similarity metrics

A similarity metric is used to compare the feature template described above to some area of an image. Section 2.5.3.3 describes the use of these similarity metrics for tracking. In this section, we describe some similarity metrics from the literature.

Burt et al. [43] give a good review and compare the computational cost of five local correlation measures in one dimension, concluding that the computationally simpler measures such as direct correlation perform nearly as well as the more complex measures on band-pass filtered images. Definitions for these correlation measures can also be found in [44]. The measures considered by Burt et al. include:

- Direct correlation:

$$C_D = \sum_{m \in N} T(m)I(i + m),$$

where  $N$  is the local neighborhood,  $T$  is the feature template,  $I$  is the image under consideration,  $T(i)$  is the value of the template at location  $i$ , and  $I(i)$  is the value of the image at location  $i$ .

- Mean normalized correlation:

$$C_M = \sum_{m \in N} (T(m) - \bar{T})(I(i+m) - \bar{I}(i)),$$

where  $\bar{T}$  is the mean value of  $T$ :

$$\bar{T} = \sum_{m \in N} T(m)/M,$$

and  $M$  is the number of elements in the neighborhood  $N$ .

- Variance normalized correlation:

$$C_V = \frac{\sum_{m \in N} (T(m) - \bar{T})(I(i+m) - \bar{I}(i))}{\sqrt{\text{Var}_T(i)} \sqrt{\text{Var}_I(i)}},$$

where the variance  $\text{Var}_T$  is given by

$$\text{Var}_T(i) = \sum_{m \in N} (T(m) - \bar{T}(i))^2,$$

and the variance  $\text{Var}_I$  is given by

$$\text{Var}_I(i) = \sum_{m \in N} (I(m) - \bar{I}(i))^2.$$

- Laplacian filtered correlation:

$$C_L = \sum_{m \in N} L_T(m) L_I(i+m),$$

where

$$L_T(i) = \sum_{m=-2}^2 w(m) T(m),$$

$$L_I(i) = \sum_{m=-2}^2 w(m) I(m),$$

and

$$w(-2) = w(2) = -0.05, \quad w(-1) = w(1) = -0.25, \quad \text{and} \quad w(0) = 0.6.$$

- Binary correlation:

$$C_B = \sum_{m \in N} B_T(m) B_I(i + m),$$

where  $B_T$  and  $B_I$  are binary images obtained by thresholding  $L_T$  and  $L_I$  at zero.

Some 1D examples are given, illustrating the benefits and drawbacks of each measure on contrived images.

Papanikolopoulos [12] uses tracking results for robotic visual servoing experiments. He uses the standard sum-of-squared-differences (SSD) metric for grayscale images:

$$SSD(T(i, j), I(x, y)) = \sum_{m, n \in N} [T(i + m, j + n) - I(x + m, y + n)]^2. \quad (2.1)$$

Singh and Allen [14] use this SSD metric for the computation of image flow.

Anandan [15] uses the SSD metric with a  $5 \times 5$  neighborhood to compute optical flow. He computes the Gaussian-weighted sum of the squared differences between the values of corresponding pixels in the template and candidate image window. That is, if  $T(i, j)$  represents the value of the pixel at location  $(i, j)$  in the template, and  $I(x, y)$  represents the value of the pixel at location  $(x, y)$  in the image,

$$SSD_{Ana}(T(i, j), I(x, y)) = \sum_{m, n \in N} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{m^2 n^2}{2}\right) [T(i + m, j + n) - I(x + m, y + n)]^2.$$

Papanikolopoulos also extends the SSD measure for use in color images, and uses the tracking results for robotic visual servoing experiments. The intensity at a pixel in a color image can be represented by a 3D vector composed of the red, blue, and green components of the color of the pixel. The vector valued function  $\mathbf{I}(x, y)$  at the pixel  $(x, y)$  of the image  $I$  is described by

$$\mathbf{I}(x, y) = (I^R(x, y), I^G(x, y), I^B(x, y))^T,$$

where  $I^R(x, y)$ ,  $I^G(x, y)$ , and  $I^B(x, y)$  are the intensities of the red, green, and blue components of the intensity of the pixel  $(x, y)$ . Papanikolopoulos defines the SSD metric for color images as

$$SSD_{Pap2}(T(i, j), I(x, y)) = \sum_{m, n \in N} \|\mathbf{T}(i + m, j + n) - \mathbf{I}(x + m, y + n)\|^2,$$

where  $\mathbf{T}$  is the vector valued intensity for the template image, defined similarly to  $\mathbf{I}$  above. The magnitude  $\|\mathbf{I}(x, y)\|$  is defined as

$$\|\mathbf{I}(x, y)\| = [(I^R(x, y))^2 + (I^G(x, y))^2 + (I^B(x, y))^2]^{1/2}.$$

Brunelli and Messelodi [45] compare three similarity metrics to the standard correlation coefficient. Their metrics are based on the  $L_1$  and  $L_2$  norms. They interpret the SSD metric as a Euclidean distance between two vectors:

$$d_2^2(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^n (x_i - y_i)^2,$$

where  $\mathbf{x} = x_1, \dots, x_n$  and  $\mathbf{y} = y_1, \dots, y_n$  represent the patterns to be compared. Since images are often normalized to zero mean and unit variance, the distance between the normalized vectors  $\mathbf{x}'$  and  $\mathbf{y}'$  can be expressed by

$$d_2^2(\mathbf{x}', \mathbf{y}') = 2n(1 - \rho_{xy}),$$

where  $\rho_{xy}$  is the correlation coefficient of the original data. The value of  $\rho_{xy}$  then represents a similarity metric between the two images. Brunelli and Messelodi propose several statistically motivated estimators for  $\rho_{xy}$ .

A commonly used estimator of  $\rho$  is the Pearson coefficient, based on the correlation of the (normalized) sample of  $\mathbf{x}$  and  $\mathbf{y}$ . This reduces in the case of image correlation to

$$r(\mathbf{x}, \mathbf{y}) = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2} \sqrt{\sum_i y_i^2}}.$$

The normalization performed above is sensitive to gross outliers, due the properties of the Euclidean norm used. Such erroneous data frequently occur in real world images, for example, due to transmission errors, specularities, and salt and pepper noise. Brunelli and Messelodi propose to use a distance metric more robust to this type of noise, the  $L_1$  norm

$$d_1(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^n |x_i - y_i|,$$

and define two similarity metrics based on this norm:

$$g(\mathbf{x}', \mathbf{y}') = 1 - \frac{\sum_i |x'_i - y'_i|}{\sum_i |x'_i| + |y'_i|}$$

$$l(\mathbf{x}', \mathbf{y}') = \frac{1}{n} \sum_i \left( 1 - \frac{|x'_i - y'_i|}{|x'_i| + |y'_i|} \right).$$

These metrics are transformed into estimators of  $\rho$  and compared to  $r(\mathbf{x}, \mathbf{y})$  above.

These metrics are then compared with a statistically robust estimator for the correlation:

$$R = \frac{Var_{th}(X + Y) - Var_{th}(X - Y)}{Var_{th}(X + Y) + Var_{th}(X - Y)},$$

where  $Var_{th}(X)$  is the tanh-estimator for the variance of the random variable  $X$ , due to Hampel [46]. The tanh-estimator is an example of the class of  $M$ -estimators introduced by Huber [47] that generalize the maximum likelihood estimator. See [48] for more explanation of the class of  $M$ -estimators in robust statistics.

These correlation estimators are chosen for their distributional robustness. The distributional robustness of an estimator is defined as insensitivity to deviations in the assumed statistical models for the data. In this case, a study was made of the behavior of these metrics in autocorrelation with injected noise. Autocorrelation is the correlation of data with itself. In the noiseless case, autocorrelation gives a maximal value for any similarity metric. The degradation of the estimators as noise is injected reveals the robustness of the estimators. The behavior of these metrics in the context of face recognition and textbook cover recognition is also studied in [45].

Brunelli and Poggio [49] compare two dissimilarity metrics to their matched spatial filtering approach: *Image Subtraction*:

$$D_{TI} = \sum_{m,n \in N} |T(m, n) - I(x + m, y + n)|,$$

and *normalized cross correlation*:

$$C_{TI} = \frac{\sum_{m,n \in N} T(m, n)I(x + m, y + n)}{\sqrt{\sum_{m,n \in N} (I(x + m, y + n))^2}}.$$

See, for example, [44] for a discussion of the use of matched filters for feature extraction.

Each of the metrics defined in this section attempts to rate the similarity of an image patch  $I$  to a template image  $T$ . Since object motion in images may introduce distortions of  $T$ , some metrics attempt to reduce the sensitivity of the similarity metric to certain types of distortions. For example, if  $I$  and  $T$  are identical except for a change in the mean value, mean normalized correlation will still rate the comparison as a good match.

Recently, researchers have been looking at ways to make similarity metric invariant to more complex distortions. Hager et al. [50] presents a local correlation metric with provisions to account for affine distortion and changes in local scene illumination, by combining information from several templates. Each template represents the same 3D feature, under different illumination conditions. The matching measure used for the individual templates is invariant to affine distortions of the feature. The match scores from these templates are combined to arrive at a final similarity metric.

Slater and Healey [51] use ideas from physics-based vision to develop a model for illumination and geometric invariant recognition of local image structure. They use a finite dimensional linear model for surface spectral reflectance to store local spatial structure information in matrix form. Illumination changes then correspond to linear transformations in this matrix, and surface rotations correspond to circular shifts of the matrices. This enables recognition of local image structure that is invariant to illumination and rotation.

Birchfield and Tomasi [52] propose a dissimilarity measure intended to be insensitive to image sampling. They use linearly interpolated intensity functions surrounding the pixels to develop the measure, which is then used in stereo matching.

### **2.5.3.3 Feature location with similarity metrics**

By using one of the similarity metrics described in Section 2.5.3.2, a template image is compared to different areas of the input image. Typically, the center of the area of the input

image having the highest similarity score (or equivalently, the lowest dissimilarity score) is taken to be the location of the feature described by the template in the input image.

A search window is typically established in the input image. In this search window, the area surrounding each pixel is compared with the template. The process of predicting feature locations described in Section 2.4 often plays a role in selecting the location and size of the search window. Singh and Allen [14] define a fixed size square search window surrounding the previous location of the feature.

Kosaka and Kak [39] consider at length the shape and location of the search window. They model the scene and compute a spatial probability density function for the location of each feature. This function determines the most likely location in the image for each feature to appear. They then search the image area corresponding to 85% of the probability mass.

Rizzi and Koditschek tracked a white ping-pong ball against a black backdrop [35], for the purpose of juggling the ping-pong ball. They use tracking information about the ball and knowledge of free-falling bodies to compute a search window for the ball.

#### 2.5.3.4 Confidence measures

It has been noted [15] that these similarity measures often lead to some unreliable matches, particularly in image regions with little textural information. For this reason, it is often helpful to compute a confidence on the match found, as well as a location. This confidence measure gives information, regarding the reliability and robustness of the match score.

Anandan [53] is generally credited with the first use of a confidence measure on the displacements generated by correlation measures in 1987. He defined the *SSD surface* (SSDS) over the space of displacements, with its height as the SSD value corresponding to each displacement. Matthies et al. [54] note that the shape of the SSDS is maintained even under significant noise corruption and that the curvature of the surface seems to be proportional to the quality of the best match in the search region. Anandan used this fact and computed the curvature of the SSDS along the four main axes at the SSDS minimum. The SSDS was considered to be

centered at the point of the best match, and the index  $(0, 0)$  corresponded to the displacement that gave the best match. Anandan defined the four normalized directional second derivatives as follows:

$$\begin{aligned} c_0 &= \frac{SSD(0, -1) - 2SSD(0, 0) + SSD(0, 1)}{SSD(0, -1) + 2SSD(0, 0) + SSD(0, 1)} \\ c_{45} &= \frac{SSD(1, -1) - 2SSD(0, 0) + SSD(-1, 1)}{SSD(1, -1) + 2SSD(0, 0) + SSD(-1, 1)} \\ c_{90} &= \frac{SSD(-1, 0) - 2SSD(0, 0) + SSD(1, 0)}{SSD(-1, 0) + 2SSD(0, 0) + SSD(1, 0)} \\ c_{135} &= \frac{SSD(-1, -1) - 2SSD(0, 0) + SSD(1, 1)}{SSD(-1, -1) + 2SSD(0, 0) + SSD(1, 1)}. \end{aligned}$$

Anandan defined the confidence measure on the match as

$$conf = \min(c_0, c_{45}, c_{90}, c_{135}).$$

Papanikolopoulos [12] notes that this confidence measure is ill-conditioned, due to the reliance on the computation of discrete second-order derivatives, and he proposes well-conditioned confidence measures on the match. He extended a 1D method of fitting parabolas to the SSD scores in the directions of the four main axes by a least-squares method developed by Matthies et al. [54], and defined  $a_0$ ,  $a_{45}$ ,  $a_{90}$ , and  $a_{135}$  as the second-order parameters (curvature) of these parabolas. Papanikolopoulos's confidence measure is then

$$CONFA = \min(a_0, a_{45}, a_{90}, a_{135}).$$

He also defined two confidence measures that capture the statistical properties of the SSDs,

$$\begin{aligned} CONFB &= \min(s_0, s_{45}, s_{90}, s_{135}) \\ s_k^2 &= \frac{1}{M_k - 1} \left[ \sum_{i,j \in N_k} SSD(i, j)^2 - M_k \overline{SSD}_k^2 \right] \\ CONFC &= \min(r_0, r_{45}, r_{90}, r_{135}) \\ r_k^2 &= \frac{1}{M_k - 1} \sum_{i,j \in N_k} (SSD(i, j) - SSD_{min,k})^2, \end{aligned}$$

where  $s_k$  and  $r_k$  values as given are the sample standard deviations in the directions of the four main axes, and the neighborhood  $N_k$  is of size  $M_k$ . The neighborhoods  $N_k$  are chosen along



the principal axis given by  $k$ . The symbol  $\overline{SSD}_k$  denotes the mean value the SSD scores for a neighborhood. These three confidence measures are then used for automatic feature selection.

Anandan [15] used the SSD matching scores of a template with a  $5 \times 5$  image region to develop a match confidence measure. This measure is based on the variation of the SSD values over the set of candidate matches. Anandan argued that if the variation of the SSD measure along a particular line in the search area surrounding the best match is small, then the component of the displacement along the direction of that line cannot be uniquely determined. Conversely, if there is significant variation along a line in the search area, the displacement along this line is more likely correct.

He found the curvatures along the *principal axes* of the surface at the point of the best match, and used this in the confidence measure. The principal axes are defined as the directions along which the curvature of the surface is extremal. The confidence measures were defined as

$$c_{max} = \frac{C_{max}}{k_1 + k_2 S_{min} + k_3 C_{max}}$$

$$c_{min} = \frac{C_{min}}{k_1 + k_2 S_{min} + k_3 C_{min}},$$

where  $k_1$ ,  $k_2$ , and  $k_3$  are normalization parameters,  $S_{min}$  is the SSD score corresponding to the best match,  $C_{max}$  is the maximal surface curvature, and  $C_{min}$  is the minimal surface curvature.

The justification for the form of the confidence measure is as follows:

- The confidence measure should be proportional to the corresponding curvature, since the reliability of the displacement should be proportional to the curvature of the SSDS.
- Since a large value for  $S_{min}$  indicates an unreliable match due to occlusion, noise, or image distortion, the confidence is inversely proportional to  $S_{min}$ .
- The presence of the term containing  $k_3$  is useful to restrict the range of the confidence measure to  $(0, 1/k_3)$ .

- The normalization factor  $k_1$  is used to maintain a finite value for the confidence measure when  $S_{min}$  tends to zero, as it will for a perfect match between a template and an image patch.
- The normalization factor  $k_2$  allows the adjustment of the relative importance assigned to the factors described above.

Anandan’s confidence measures are used in a hierarchical framework to compute dense displacement fields for image sequences.

Singh and Allen [14] define a *response distribution* based on the SSD metric Equation (2.1) as

$$\mathcal{RD}_c(u, v) = \exp(-kSSD(u, v)), \quad (2.2)$$

where  $k$  is used as a normalization factor. The normalization factor  $k$  was chosen in this work so that the maximum response was 0.95. Singh and Allen argue for a probabilistic interpretation of the response surface. Each point in the search area is a candidate for the “true match.” However, a point with a small response is less likely to be the true match than a point with a high response. Thus, the response distribution could be interpreted as a probability distribution on the true match location – the response at a point depicting the likelihood of the corresponding match being the true match. This interpretation of the response distribution allows the use of estimation-theoretic techniques. These techniques are used to compute the image flow at a point and to associate a confidence value with this flow.

Instead of taking the minimum SSD score as most likely location of the match, Singh and Allen [14] obtain an estimate of the location of the best match by using a weighted least-squares

approach:

$$u_m = \frac{\sum_{u,v \in N} \mathcal{RD}(u,v)u}{\sum_{u,v \in N} \mathcal{RD}(u,v)}$$

$$v_m = \frac{\sum_{u,v \in N} \mathcal{RD}(u,v)v}{\sum_{u,v \in N} \mathcal{RD}(u,v)},$$

and, under the assumption of additive zero mean independent errors, associate a covariance matrix with this estimate:

$$\mathbf{P}_m = \begin{bmatrix} \frac{\sum_{u,v \in N} \mathcal{RD}(u,v)(u - u_m)^2}{\sum_{u,v \in N} \mathcal{RD}(u,v)} & \frac{\sum_{u,v \in N} \mathcal{RD}(u,v)(u - u_m)(v - v_m)}{\sum_{u,v \in N} \mathcal{RD}(u,v)} \\ \frac{\sum_{u,v \in N} \mathcal{RD}(u,v)(u - u_m)(v - v_m)}{\sum_{u,v \in N} \mathcal{RD}(u,v)} & \frac{\sum_{u,v \in N} \mathcal{RD}(u,v)(v - v_m)^2}{\sum_{u,v \in N} \mathcal{RD}(u,v)} \end{bmatrix}. \quad (2.3)$$

The reciprocals of the eigenvalues of the covariance matrix are used as confidence measures associated with the estimate, along the directions given by the corresponding eigenvectors. To our knowledge, this is the first instance where the location of the best match is treated as a random vector, and the (normalized) SSD surface is used to compute the spatial certainty of the estimate of this vector. These confidence measures are used in the propagation of high confidence measurements for image flow to regions with lower confidence measurements, such as caused by large homogeneous regions.

It is also possible to model the tracking results with a probability distribution and to use methods from robust statistics to reject tracking results that are not modeled well by this distribution. Tommasini et al. [55] use this approach to make point-tracking results more robust to occlusion.

## 2.6 Object Tracking

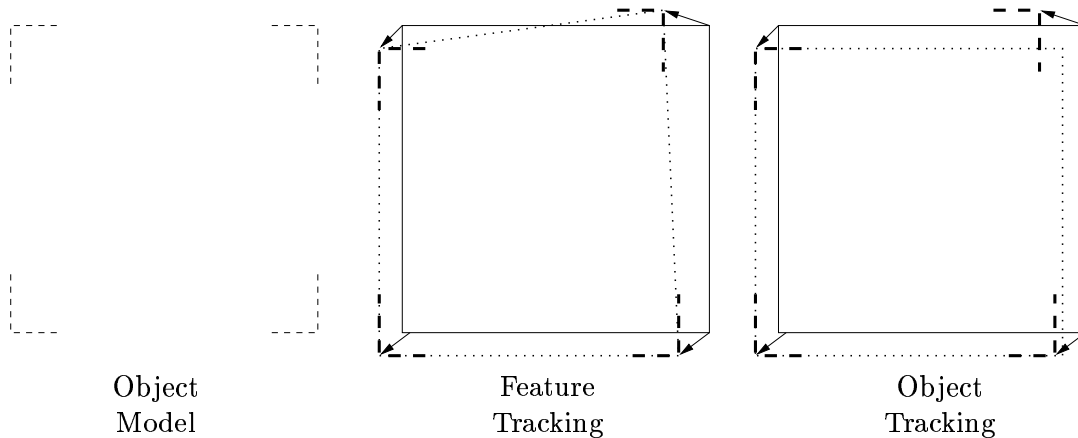
*Object tracking* can be defined as updating the configuration, in the image plane or in three space, of an object model. An object model can be a simple 2D shape model, or can contain more complex information about the object, such as the appearance of portions of the object or the locations on the object of interesting features.

Feature tracking, as defined above, is commonly used to aid in object tracking. However, object tracking as we have defined it requires some additional interpretation in addition to feature tracking. There must be some object model whose configuration is being tracked. In the degenerate case, the object is a particle, and the choices for object models are as described above.

More commonly, 3D rigid motion is extracted by analyzing the tracking results of these point features. This is a well-established area of research. An overview of this research can be found in [56].

Gennery [10] tracks rigid polyhedral objects. His object model includes a wire frame of the object and constant reflectivity coefficients for each face of the object. His dynamic model includes the position, velocity, and acceleration of the object. Bray [38] also tracks polyhedral objects, utilizing a similar object model. Lowe [26] tracks objects with internal degrees of freedom, for example, a box with one degree of freedom (a hinged lid). The object model is a wire frame model, with parameters for the position and orientation of the base and an extra parameter for the angle between the box body and lid. Not all object models must have a geometric interpretation, however. Stephens [37] tracks rigid polyhedral objects using a local Hough transform, so his object model is a point in the six-dimensional Hough space.

All these systems have the unifying characteristic that there is an underlying object model, and results from the tracking of features are used to update the object parameters. One advantage to this extra modeling is the ability to treat features as items of secondary interest, and track the object in spite of the occlusion of some features. Therefore, if an individual feature tracker fails but other trackers succeed in localizing their features, the object will still



**Figure 2.1** The use of object models.

be tracked. Hashimoto et al. [57] have shown that increasing the number of redundant features increases the robustness of visual servoing.

By assuming some relationship between the locations of the features, such as the features belonging to a rigid object in the plane, or the features acting as control points on a 2D contour, the position and configuration of that object is estimated, leading to predictions about the feature point locations, rather than predicting the locations separately. Figure 2.1 shows this idea, for the case of an object model consisting of four corners. If feature tracking alone is used (no object model assumed), mis-tracking of the corner features results in a warped tracked object. If the relationship of the corners to each other are used to constrain the tracking results, the object can be tracked in spite of one corner feature being mis-tracked.

Note that the object model and dynamic models discussed above can be used in almost any combination. In some situations, it is possible and often reasonable, for example, to have a complicated object model, but not to include any dynamic model at all [11]. In certain other situations, it is reasonable to include a more complex dynamic model for the object, but to consider the object to be a particle (i.e., no object model) [35].

## 2.7 Kalman Filtering in Feature and Object Tracking

The late 1950s were a time of great innovation in control theory. First, the *maximum principle*, a set of necessary conditions for the optimality of an open loop control policy, was developed. Second, *dynamic programming*, the systematic use of intermediate partial programming results to solve larger problems, was popularized. This laid the groundwork for an influential development, *Kalman filtering*. Kalman [58] provided a breakthrough in the extension of Wiener filtering, with its assumptions of stationarity, to nonstationary processes with state-space descriptions. Kalman filtering provided a mathematical theory for recursive estimation and predication of an unknown time function on the basis of another, observed one [59].

Kalman filtering was immediately appealing to the control community for a number of reasons. A solution to the linear system was obtained in a convenient recursive form requiring the off-line solution of a Ricatti (differential) equation [60]. This allowed system analysts to formulate their problems in *state space*, as well as the classical spectral factorization approach required by Wiener filtering. The Kalman filtering approach also provided an algorithm suitable for almost immediate implementation on a digital computer. In the community as a whole, the use of state models was taking off. This resulted in the creation of a separate field of study investigating the construction of these models, that Kalman termed *realization theory*.

Both the continuous time version [61] (the Kalman-Bucy filter) and the discrete time version [58] (the Kalman filter) found immediate applications. The beginning of the space age with the launching of Sputnik in 1957 introduced many applications for this theory. Engineers at NASA, the Draper Laboratories of MIT, and elsewhere began using the Kalman-Bucy filter in control applications [62].

By the 1980s, Kalman filtering was beginning to be used in system analysis. Hallam [63] uses a standard Kalman filter to track several sonar targets. This tracking is used in the context of underwater robotic navigation and is used to determine the observer motion. This is a slight extension of the historical use of the Kalman filter's use in radar tracking applications [64] in that radar tracking has no observer motion to extract.

Ayache and Faugeras [65] utilize the least-squares optimality of Kalman filtering (see Chapter 3) to estimate the orientation of a 2D model based on the location and orientation of multiple line segments belonging to that model.

Coelho and Nunes [66] present an example of the use of the EKF to calibrate the joint angles of a PUMA (Programmable Universal Machine for Assembly) robotic arm. In this application, the 3D position of the end-effector base is taken as an input, and the true joint angles of the first three joints of the arm are desired. The forward kinematics of the arm are modeled in the filter. The initial (assumed) position of the end-effector base is assumed to be “sufficiently close” to the actual position, so that the linearizations inherent in the EKF are reasonably accurate. The filter is shown to converge to the “true” joint angles, given the actual 3D position of the end-effector base. This is one of the earlier examples of the use of the EKF to model and functionally invert, in a local area, the forward kinematics of a robotic arm. Note that for a nonlinear system such as a robotic arm this inversion is only possible in the local area around a given state.

By the late 1980s the influence of the Kalman filter had spread, and it was being applied in increasingly complex situations. One of the earliest reports of the use of a least squares spatiotemporal filter for the visual tracking of known 3D objects is by Gennery in 1982 [67]. Gennery does not use the standard Kalman filter, but derives a similar approach based on direct analysis of the covariance of the observed data.

At this point in the development of object tracking algorithms, there were two main approaches, depending on the measurement data used [13]. One is based on the optical flow field, and is describe in Section 2.1. The other is based on the correspondence of discrete features such as lines, points, and contours [68], [69]. Nagel gives a review of some applications of correspondence-based techniques in use in 1983 in [56]. To this point, work on image sequences typically used two or three frames of noisy images. By summarizing tracking history in a finite dimensional *state vector*, researchers were able to use longer image sequences for tracking. This avoids the nonrobustness and lack of numerical stability inherent in two and three view monoc-

ular image based algorithms [13]. The Kalman filter began to be used as a systems framework for these tracking algorithms.

Although many researchers are using an established framework based in systems theory, there is still work going on in direct modeling and analysis. Lowe [26] typifies more recent work in this area, focusing on the analysis of uncertainty and the propagation of error through the forward kinematics of an object, and the projection of the image of an object onto a 2D sensor. Huttenlocher et al. [70] decompose the motion of 3D nonrigid objects into two components: a 2D shape change and a 2D motion of the object image in the sensor plane and analyze each component separately, enabling the use of distance metrics appropriate to the specific component. Gee and Cippolla [71] approach the tracking problem as one of pose estimation on a minimal subset of the observed data and use robust regression based on Bayesian inference to select the best subset of the data to use for the pose estimate.

In the late 1980s, researchers began directly using Kalman filtering to estimate object state by modeling the system structure. Faugeras et al. [72] propose the use of the Kalman filter for integrating noisy stereo measurements to analyze the 3D structure, but give no experimental results. In [73], Broida and Chellappa employ a Kalman filter to estimate the 2D rotation and 1D translation of an object from a sequence of simulated 1D images. Matthies and Shafer [74] describe the use of several landmark points to estimate the location of a mobile robot using Kalman filtering.

Dickmanns and Graefe [75] presented a seminal work describing the relationship of the modeled world (object state) to the real world and measured world (feature measurements) and describing the use of the framework in guiding autonomous vehicles [76]. Since then, this dynamic machine vision approach has been used in many situations, and the use of Kalman filtering to implicitly invert the complex photogrammetric equations relating image plane feature measurements to objects in the world has increased [3]. Wu et al. [77], use a small number of point correspondences to track the motion of a rigid object moving with one degree of translational motion and one degree of rotational freedom, about an unknown axis of rotation.



Andersen et al. [78] exploit the dynamic modeling capacity of the Kalman filter to estimate the state of a ball bearing rolling on a wooden board. The board is part of a labyrinth game and is mounted on gimbals. The object of the game is to move the ball through a maze without falling into holes in the board. Allen et al. [4] and Lee et al. [79] use different state formulations in Kalman filters to track a toy train moving on an oval track with an end-effector mounted camera. Allen et al. parameterize the motion of the train in a local coordinate system to better achieve the whiteness assumptions made by the EKF (see Section 3.2 for details on these assumptions). Lee et al. use a relative coordinate system with respect to the camera, including the linear and rotational velocities of the target, the surface normal vector of the target, the angle between the surface normal and visual direction, and the tilt of the surface tangent plane in the object state. Wilson et al. [3] report the use of extended Kalman filtering to estimate the relative POSE (position and orientation) of a rigid work piece with respect to the camera, for the purpose of specifying a desired POSE and moving the camera appropriately. Up to five corner or hole features are used to determine the POSE of the work piece.

In addition to its use as an algorithm for incorporating feature measurements into a state estimate, many researchers have benefited from the use of the Kalman filter as a purely spatiotemporal filter. In this way, spurious measurements and noisy data can be handled in a numerically robust manner. The filtered measurements can then be used in an analysis of the scene to generate more robust estimates for object states. Matthies et al. [54] estimate the depth of each pixel in an image via an optical flow approach, using a Kalman filters to smooth changes in the depth map at each pixel. This results in a dense depth estimate over the image. Singh and Allen [14] estimate the depth field in an image in a similar manner, but make more sophisticated use of confidence measurements on the depth field to propagate depth estimates from regions of high certainty to regions with uncertain estimates. Wunsch and Hirzinger [80] propose an efficient model-based pose estimation method, then use a Kalman filter to smooth the pose estimates. Metaxas and Terzopoulos [81] pose the tracking problem as one of physics based synthesis, and use the continuous Kalman-Bucy filter to synthesize and smooth the non-

rigid shape of an object deforming and moving under external applied forces and imposed constraints. Blake et al. [82] use a Kalman filtering framework to help in the tracking of an active contour. Matteuci et al. [5] track the bounding rectangle of a vehicle moving along an assumed ground plane, then filter the 2D motion estimates using a simple dynamic model for the vehicle. Stark and Fuchs [83] use an active contour model with a Kalman filter to track the 2D silhouette of an object in an image sequence, then use points on the contour to infer the 3D pose of a known object.

## CHAPTER 3

### KALMAN FILTERING

There are many ways to estimate an unknown quantity from available data. One class of estimators, known as stochastic estimators, models the deviation of the measurements of a process from the actual output of that process as random variables. Often, the underlying process is modeled as a random process, but that is not necessary.

In this chapter, we derive the classical Kalman filter equations for a vector valued unknown quantity. We then derive the extended Kalman filter by linearizing the system dynamics about the current state estimate. The reader familiar with extended Kalman filtering may wish to refer to Equations (3.25)-(3.31) for the notation used in this thesis. In Chapter 6, we will use the extended Kalman filter derived in this chapter.

Possibly the simplest way to estimate an unknown vector valued  $\mathbf{x}$  from observed vector data  $\mathbf{z}$  is *mean-square estimation*, where the estimate  $\hat{\mathbf{x}}$  is chosen to minimize the expected value of the Euclidean norm squared of the error  $E[(\mathbf{x} - \hat{\mathbf{x}})^T(\mathbf{x} - \hat{\mathbf{x}})]$ . This can easily be extended to estimate functions of the quantity  $\mathbf{x}$ . The Kalman filter implements a recursive least squares fit to the data, given some assumptions about the system.

Our derivation will proceed as follows. First, we will decompose a general linear system into stochastic and deterministic subsystems. Then, we will find a least squares estimate for the state of the system at time  $k$  ( $\hat{\mathbf{x}}_{k|k-1}$ ) in terms of the observed data at time  $k$  ( $\mathbf{z}_k$ ). Next we will use this result to find a least squares estimate for the state of the system at time  $k$  ( $\hat{\mathbf{x}}_{k|k}$ ) in terms of all the observed data ( $\mathbf{z}_0, \dots, \mathbf{z}_j, j \leq k$ ). We will then derive a recursive formulation of

the same computation. We combine the recursive solutions for the stochastic subsystem with the well-known solution for the deterministic subsystem to arrive at the classical Kalman filter equations. These equations are used to extend the estimation theory to the nonlinear case, resulting in the extended Kalman filter (EKF). Finally, an algebraically equivalent form of the EKF is derived that has some computational advantages in our application.

## 3.1 Basic Kalman Filtering

### 3.1.1 System definition

We describe a linear plant with noise, like the one shown in Figure 3.1(a) by a discrete time *state model* of the following form:

$$\begin{aligned} \underbrace{\mathbf{x}_{k+1}}_{n \times 1} &= \underbrace{\mathbf{A}_k}_{n \times n} \underbrace{\mathbf{x}_k}_{n \times 1} + \underbrace{\mathbf{B}_k}_{n \times m} \underbrace{\mathbf{u}_k}_{m \times 1} + \underbrace{\mathbf{G}_k}_{n \times p} \underbrace{\mathbf{w}_k}_{p \times 1} \\ \underbrace{\mathbf{z}_k}_{q \times 1} &= \underbrace{\mathbf{H}_k}_{q \times n} \underbrace{\mathbf{x}_k}_{n \times 1} + \underbrace{\mathbf{D}_k}_{q \times m} \underbrace{\mathbf{u}_k}_{m \times 1} + \underbrace{\mathbf{v}_k}_{p \times 1} \end{aligned} \tag{3.1}$$

Both the *process noise*,  $\mathbf{w}_k$ , and *measurement noise*,  $\mathbf{v}_k$ , are assumed to be sequences of zero-mean Gaussian white noise such that  $Var(\mathbf{w}_k) = \mathbf{Q}_k$ <sup>1</sup> and  $Var(\mathbf{v}_k) = \mathbf{R}_k$  are positive definite matrices, and  $E(\mathbf{w}_k \mathbf{v}_l^T) = 0$  for all  $k$  and  $l$ . See Table 3.1 for the matrix dimensions.

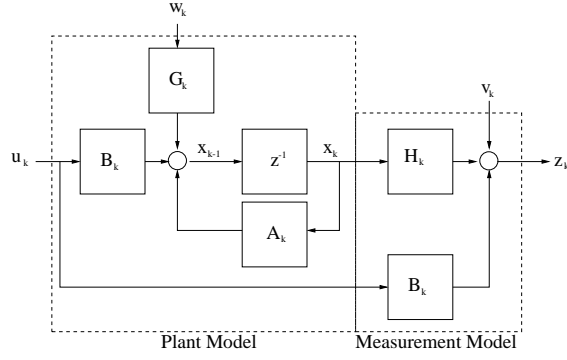
In a physical system, the *state* can be any set of relevant parameters. Formally, relevant parameters are defined as those parameters needed to uniquely determine the output of a system, given the input to the system. For example, in a robotic arm, the state might be the configuration of the robot. A configuration of an object is a set of numbers that give a specification of the position of every point on the object. Configuration space is defined to be the space of all possible configurations of a object.

### 3.1.2 Decomposition of the system

We can decompose this linear system into two disjoint systems,<sup>2</sup> one composed of the deterministic elements of Equation (3.1) (see Figure 3.1(c)) and one composed of the stochastic

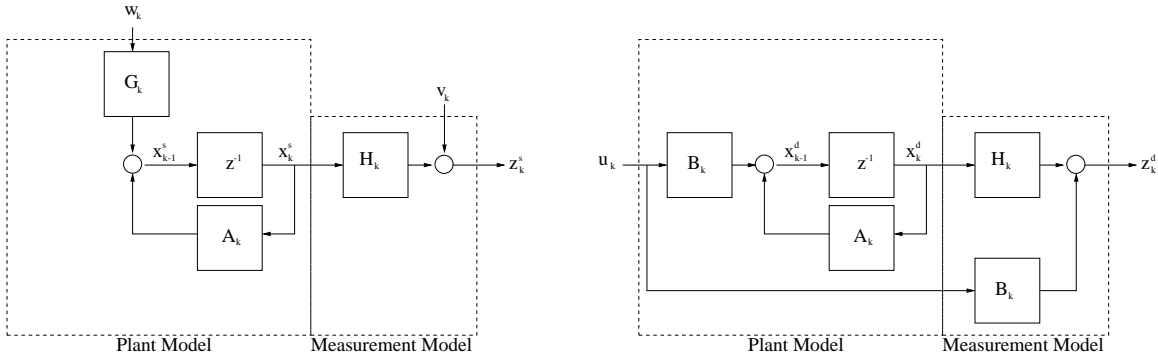
<sup>1</sup>The variance of a vector is simply the covariance matrix of the vector with itself ( $Var(\mathbf{v}) = Cov(\mathbf{v}, \mathbf{v})$ ).

<sup>2</sup>This derivation follows closely that of [84].



a) General System Model =

b) Stochastic System Model + c) Deterministic System Model



**Figure 3.1** Plant and measurement models.

elements (see Figure 3.1(b)). Then Equation (3.1) is the direct sum of these two subsystems. This has the advantage that the deterministic part has a well-known solution, simplifying the derivation below. We will combine the solution derived below with the deterministic solution to arrive in Section 3.1.5 at the Kalman filter.

As shown in Figure 3.1, the general system given in Equation (3.1) can be decomposed into the sum of a purely stochastic system

$$\begin{aligned}
 \mathbf{x}_{k+1}^s &= \mathbf{A}_k \mathbf{x}_k^s + \mathbf{G}_k \mathbf{w}_k \\
 \mathbf{z}_k^s &= \mathbf{H}_k \mathbf{x}_k^s + \mathbf{v}_k,
 \end{aligned}
 \tag{3.2}$$

**Table 3.1** Notation used in this section.

Symbol	Definition	Dimension	Notes
$\mathbf{x}_k$	state (at the $k$ th time instant)	$n \times 1$	$n \geq 1$
$\mathbf{A}_k$	system matrices	$n \times n$	
$\mathbf{B}_k$	control input matrices	$n \times m$	$1 \leq m \leq n$
$\mathbf{u}_k$	deterministic control input	$m \times 1$	
$\mathbf{G}_k$	system noise matrices	$n \times p$	$1 \leq p \leq n$
$\mathbf{w}_k$	white process noise	$p \times 1$	
$\mathbf{Q}_k$	covariance for $w_k$	$p \times p$	
$\mathbf{z}_k$	measurement	$q \times 1$	$1 \leq q \leq n$
$\mathbf{H}_k$	measurement matrices	$q \times n$	
$\mathbf{v}_k$	white measurement noise	$q \times 1$	
$\mathbf{R}_k$	covariance for $v_k$	$q \times q$	
$\bar{\mathbf{z}}_k$	measurement history	$kq \times 1$	
$\mathbf{C}_{k,j}$	measurement/system matrix history	$jq \times n$	$j \leq k$
$\bar{\mathbf{e}}_{k,j}$	noise propagation matrices	$jq \times 1$	
$\Phi_{j,k}$	transition matrices	$n \times n$	
$\mathbf{e}_{k,l}$	noise matrices	$q \times 1$	
$\mathbf{W}_{k,k}$	weighting matrix	$kq \times kq$	
$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_{k k}$	state estimate at time $k$ given data $\mathbf{z}_0 \dots \mathbf{z}_k$	$n \times 1$	
$\hat{\mathbf{x}}_{k k-1}$	state estimate at time $k$ given data $\mathbf{z}_0 \dots \mathbf{z}_{k-1}$	$n \times 1$	

and a purely deterministic system

$$\begin{aligned} \mathbf{x}_{k+1}^d &= \mathbf{A}_k \mathbf{x}_k^d + \mathbf{B}_k \mathbf{u}_k \\ \mathbf{z}_k^d &= \mathbf{H}_k \mathbf{x}_k^d + \mathbf{D}_k \mathbf{u}_k, \end{aligned} \tag{3.3}$$

where  $\mathbf{x}_k = \mathbf{x}_k^s + \mathbf{x}_k^d$ , and  $\mathbf{z}_k = \mathbf{z}_k^s + \mathbf{z}_k^d$ . Note that the transformation matrices  $\mathbf{H}_k$  and  $\mathbf{A}_k$  appear in both the stochastic and deterministic subsystem. A reduction in the number of states is possible, but we will assume none is done to simplify the presentation, so each of  $\mathbf{x}_k$ ,  $\mathbf{x}_k^s$ , and  $\mathbf{x}_k^d$  are  $n \times 1$  vectors. Similarly, each of  $\mathbf{z}_k$ ,  $\mathbf{z}_k^s$ , and  $\mathbf{z}_k^d$  are  $q \times 1$  vectors. To simplify the notation, we will omit the superscripts unless there is a possibility of confusion between the stochastic and deterministic systems.

### 3.1.3 Solution for stochastic system

In this section, we derive a least square estimate for the state of the stochastic system given in Equation (3.2) at time  $k$ ,  $\hat{\mathbf{x}}_k$ , first in terms of only the data at time  $k$ ,  $\mathbf{z}_k$ , then in terms of all the observed data  $\mathbf{z}_0, \dots, \mathbf{z}_j, j \leq k$ .

### 3.1.3.1 No history

The solution for a linear system in terms of the output of that system is well known from linear systems theory. This solution will be combined with Proposition 2 (below) to derive a least squares estimate for our particular linear system. The reader familiar with weighted least squares may wish to skip this proposition.

**Proposition 1 (optimal least squares for a linear system):** *Given the observation equation of a linear system where the observations  $\mathbf{z}_k$  are corrupted by a zero mean white noise sequence  $\mathbf{v}_k$ ,*

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k,$$

*the optimal estimate (in the least squares sense) of  $\mathbf{x}_k$  using the data  $\mathbf{z}_k$  is:*

$$\hat{\mathbf{x}}_k = (\mathbf{H}_k^T \mathbf{R}_k^{-1} \mathbf{H}_k)^{-1} \mathbf{H}_k^T \mathbf{R}_k^{-1} \mathbf{z}_k, \quad (3.4)$$

*where  $\mathbf{R}_k$  is the variance of  $\mathbf{v}_k$ ,  $\text{Var}(\mathbf{v}_k)$ .*

**Proof:** It is well known (see [84]) that for an over determined set of linear equations,  $\mathbf{b} = \mathbf{A}\mathbf{x}$ , the least-squares solution is given by:

$$\hat{\mathbf{x}} = (\mathbf{A}^T \mathbf{W} \mathbf{A})^{-1} \mathbf{A}^T \mathbf{W} \mathbf{b},$$

where  $\mathbf{W}$  is a weighting matrix among the terms in  $\mathbf{x}$ .  $\mathbf{W}$  may be set to  $\mathbf{I}$  if equal weighting among the terms in  $\mathbf{x}$  is desired. If we substitute  $\mathbf{H}_k$  for  $\mathbf{A}$  and  $\mathbf{z}_k - \mathbf{v}_k$  for  $\mathbf{b}$  in Equation (3.4), we arrive at

$$\hat{\mathbf{x}}_k = (\mathbf{H}_k^T \mathbf{W}_k \mathbf{H}_k)^{-1} \mathbf{H}_k^T \mathbf{W}_k (\mathbf{z}_k - \mathbf{v}_k).$$

Since  $\mathbf{v}_k$  is a zero-mean white noise sequence, it does not affect the mean of the estimator and therefore drops out when we solve for an unbiased estimator<sup>3</sup> for  $\mathbf{x}_k$ . The intuitive choice for  $\mathbf{W}_k$ , to weight the residuals in inverse proportion to the variance of the errors ( $\mathbf{W}_k = \text{Var}(\mathbf{v}_k)^{-1}$ ), can be shown [84] to achieve the minimum variance error, giving us the desired result. □

---

<sup>3</sup>An unbiased estimator is one for which  $E(\mathbf{x} - \hat{\mathbf{x}}) = \mathbf{0}$ .

### 3.1.3.2 Solution with observation history

We would like to find an estimate  $\hat{\mathbf{x}}_k$  of  $\mathbf{x}_k$  that incorporates all the observations  $\mathbf{z}_0, \mathbf{z}_1, \dots, \mathbf{z}_k$ . To this end, we introduce the vectors  $\bar{\mathbf{z}}_k = [\mathbf{z}_0 \dots \mathbf{z}_k]^T$  and obtain  $\hat{\mathbf{x}}_k$  from the vector  $\bar{\mathbf{z}}_k$ . We then manipulate Equation (3.2) into a form to which we can apply the well-known linear least squares solution.

For the purposes of this derivation, we will assume all system matrices  $\mathbf{A}_j$  to be nonsingular, although the Kalman filter equations can be derived without making this assumption (see [84]). With this assumption, we can achieve the result shown in Proposition 2. In addition to the usefulness of the proposition for solving the system equations in a least squares manner, it is intuitively pleasing, as a separation of the nominal system operation (expressed below as  $\mathbf{C}_{k,j}\mathbf{x}_k$ ) from the noise that has entered the system and been “fed through” the system dynamics ( $\bar{\mathbf{e}}_{k,j}$ ).

**Proposition 2:** *The output equation from Equation (3.2) can be written as follows, for  $j \leq k$ :*

$$\bar{\mathbf{z}}_j = \mathbf{C}_{k,j}\mathbf{x}_k + \bar{\mathbf{e}}_{k,j}, \quad (3.5)$$

where  $\mathbf{C}_{k,j} = \begin{bmatrix} \mathbf{H}_0\Phi_{0,k} \\ \vdots \\ \mathbf{H}_j\Phi_{j,k} \end{bmatrix}$  and  $\bar{\mathbf{e}}_{k,j} = \begin{bmatrix} \mathbf{e}_{k,0} \\ \vdots \\ \mathbf{e}_{k,j} \end{bmatrix}$ . The transition matrices  $\Phi_{j,k}$  are defined as follows:

$$\Phi_{j,k} = \begin{cases} \mathbf{A}_{j-1}\mathbf{A}_{j-2}\cdots\mathbf{A}_{k+1}\mathbf{A}_k & \text{if } j > k \\ \mathbf{I} & \text{if } j = k \\ \Phi_{k,j}^{-1} & \text{if } j < k, \end{cases} \quad (3.6)$$

and

$$\mathbf{e}_{k,l} = \mathbf{v}_l - \mathbf{H}_l \sum_{i=l+1}^k \Phi_{l,i} \mathbf{G}_{i-1} \mathbf{w}_{i-1}$$



**Proof:** Substituting the definitions of  $\mathbf{C}_{k,j}$  and  $\bar{\mathbf{e}}_{k,j}$  into Equation (3.5), the equality to prove becomes

$$\begin{bmatrix} \mathbf{H}_0 \mathbf{x}_0 + \mathbf{v}_0 \\ \vdots \\ \mathbf{H}_j \mathbf{x}_j + \mathbf{v}_j \end{bmatrix} = \begin{bmatrix} \mathbf{H}_0 \Phi_{0,k} \\ \vdots \\ \mathbf{H}_j \Phi_{j,k} \end{bmatrix} \mathbf{x}_k + \begin{bmatrix} \mathbf{v}_0 - \mathbf{H}_0 \sum_{i=1}^k \Phi_{0,i} \mathbf{G}_{i-1} \mathbf{w}_{i-1} \\ \vdots \\ \mathbf{v}_j - \mathbf{H}_j \sum_{i=j+1}^k \Phi_{j,i} \mathbf{G}_{i-1} \mathbf{w}_{i-1} \end{bmatrix}$$

and we see that it is sufficient to show that

$$\mathbf{H}_j \mathbf{x}_j + \mathbf{v}_j = \mathbf{H}_j \Phi_{j,k} \mathbf{x}_k + \mathbf{v}_j - \mathbf{H}_j \sum_{i=j+1}^k \Phi_{j,i} \mathbf{G}_{i-1} \mathbf{w}_{i-1}, \quad (3.7)$$

for  $j \leq k$ . Subtracting  $\mathbf{v}_j$  from both sides of Equation (3.7) and observing that Equation (3.7) is true for all  $\mathbf{x}_k, \mathbf{x}_j$ , and  $\mathbf{w}_i$  (thus we can drop the  $\mathbf{H}_j$ ), we can simplify Equation (3.7) to obtain

$$\mathbf{x}_j = \Phi_{j,k} \mathbf{x}_k - \sum_{i=j+1}^k \Phi_{j,i} \mathbf{G}_{i-1} \mathbf{w}_{i-1}.$$

Now we left-multiply by  $\Phi_{j,k}^{-1} = \Phi_{k,j}$ . □

**Proposition 3 (least squares estimate for  $\mathbf{x}_k$ ):** *The linear, unbiased, minimum variance least-squares estimate of  $\mathbf{x}_k$  using the data  $\mathbf{z}_0, \dots, \mathbf{z}_j$ , which we will denote as  $\hat{\mathbf{x}}_{k|j}$ , is*

$$\hat{\mathbf{x}}_{k|j} = (\mathbf{C}_{k,j}^T \mathbf{W}_{k,j} \mathbf{C}_{k,j})^{-1} \mathbf{C}_{k,j}^T \mathbf{W}_{k,j} \bar{\mathbf{z}}_j. \quad (3.8)$$

**Proof:** This follows directly from the combination of Equation (3.5) with the previous result Equation (3.4). □

### 3.1.4 Recursive solution for stochastic system

Unfortunately,  $\bar{\mathbf{z}}_j$  as derived in the previous section contains all the observations from time 0. This storage requirement can become onerous in long-running applications. In this section we will derive a recursive form of Equation (3.8), which gives  $\hat{\mathbf{x}}_{k|k}$  in terms of the previously computed quantities  $\hat{\mathbf{x}}_{k|k-1}$  and  $\hat{\mathbf{x}}_{k-1|k-1}$ .

### 3.1.4.1 Observation update

In this section, we wish to relate  $\hat{\mathbf{x}}_{k|k}$ , the state estimate given all the data  $\mathbf{z}_0, \dots, \mathbf{z}_k$ , to the previously computed estimate  $\hat{\mathbf{x}}_{k|k-1}$ , given only the data  $\mathbf{z}_0, \dots, \mathbf{z}_{k-1}$ . This relationship is commonly referred to as the observation update.

The first step is to manipulate the expression for the weighting matrix above to obtain the classical Kalman gain, which will quickly lead to the desired observation update equation. First, we observe that due to the statistical independence of  $\mathbf{v}_i$  and  $\mathbf{w}_i$ , the expression for  $\mathbf{W}_{k,k-1}^{-1}$  can be expanded:

$$\begin{aligned}
\mathbf{W}_{k,k-1}^{-1} &= \text{Var}(\bar{\mathbf{e}}_{k,k-1}) \\
&= \text{Var} \begin{bmatrix} \mathbf{e}_{k,0} \\ \vdots \\ \mathbf{e}_{k,k-1} \end{bmatrix} \\
&= \text{Var} \begin{bmatrix} \mathbf{v}_0 - \mathbf{H}_0 \sum_{i=1}^k \Phi_{0,i} \mathbf{G}_{i-1} \mathbf{w}_{i-1} \\ \vdots \\ \mathbf{v}_{k-1} - \mathbf{H}_{k-1} \Phi_{k-1,k} \mathbf{G}_{k-1} \mathbf{w}_{k-1} \end{bmatrix} \\
&= \text{Var} \begin{bmatrix} \mathbf{v}_0 \\ \vdots \\ \mathbf{v}_{k-1} \end{bmatrix} + \text{Var} \begin{bmatrix} \mathbf{H}_0 \sum_{i=1}^k \Phi_{0,i} \mathbf{G}_{i-1} \mathbf{w}_{i-1} \\ \vdots \\ \mathbf{H}_{k-1} \Phi_{k-1,k} \mathbf{G}_{k-1} \mathbf{w}_{k-1} \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{R}_0 & & 0 \\ & \ddots & \\ 0 & & \mathbf{R}_{k-1} \end{bmatrix} + \text{Var} \begin{bmatrix} \mathbf{H}_0 \sum_{i=1}^k \Phi_{0,i} \mathbf{G}_{i-1} \mathbf{w}_{i-1} \\ \vdots \\ \mathbf{H}_{k-1} \Phi_{k-1,k} \mathbf{G}_{k-1} \mathbf{w}_{k-1} \end{bmatrix}.
\end{aligned}$$

We will also find recursive forms for the matrix  $\mathbf{W}_{k,k}^{-1}$  and the matrix  $\mathbf{C}_{k,j}$  useful:

$$\underbrace{\mathbf{W}_{k,j}^{-1}}_{jq \times jq} = \text{Var}(\bar{\mathbf{e}}_{k,j}) = \begin{matrix} \underbrace{(j-1)q \times (j-1)q} & \\ \left[ \begin{array}{cc} \mathbf{W}_{k,j-1}^{-1} & 0 \\ 0 & \mathbf{R}_j^{-1} \end{array} \right] & \underbrace{q \times q} \end{matrix}, \quad (3.9)$$

$$\underbrace{\mathbf{C}_{k,j}}_{jq \times n} = \left\{ \begin{array}{l} \mathbf{C}_{k,j-1} \\ \mathbf{H}_j \end{array} \right\} \begin{matrix} (j-1)q \times n \\ q \times n \end{matrix}. \quad (3.10)$$

We now expand Equation (3.8) in terms of these recursive forms:

$$\begin{aligned} \hat{\mathbf{x}}_{k|j} &= (\mathbf{C}_{k,j}^T \mathbf{W}_{k,j} \mathbf{C}_{k,j})^{-1} \mathbf{C}_{k,j}^T \mathbf{W}_{k,j} \bar{\mathbf{z}}_j \\ &= \left( \left[ \begin{array}{cc} \mathbf{C}_{k,j-1}^T & \mathbf{H}_j^T \end{array} \right] \left[ \begin{array}{cc} \mathbf{W}_{k,j-1} & 0 \\ 0 & \mathbf{R}_j^{-1} \end{array} \right] \left[ \begin{array}{c} \mathbf{C}_{k,j-1} \\ \mathbf{H}_j \end{array} \right] \right)^{-1} \left[ \begin{array}{cc} \mathbf{C}_{k,j-1}^T & \mathbf{H}_j^T \end{array} \right] \left[ \begin{array}{cc} \mathbf{W}_{k,j-1} & 0 \\ 0 & \mathbf{R}_j^{-1} \end{array} \right] \left[ \begin{array}{c} \bar{\mathbf{z}}_{j-1} \\ \mathbf{z}_j \end{array} \right] \\ &= \left( \mathbf{C}_{k,j-1}^T \mathbf{W}_{k,j-1} \mathbf{C}_{k,j-1} + \mathbf{H}_j^T \mathbf{R}_j^{-1} \mathbf{H}_j \right)^{-1} \left( \mathbf{C}_{k,j}^T \mathbf{W}_{k,j-1} \bar{\mathbf{z}}_{j-1} + \mathbf{H}_j^T \mathbf{R}_j^{-1} \mathbf{z}_j \right) \end{aligned}$$

which can be rearranged to give

$$\left( \mathbf{C}_{k,j-1}^T \mathbf{W}_{k,j-1} \mathbf{C}_{k,j-1} + \mathbf{H}_j^T \mathbf{R}_j^{-1} \mathbf{H}_j \right) \hat{\mathbf{x}}_{k|j} = \left( \mathbf{C}_{k,j}^T \mathbf{W}_{k,j-1} \bar{\mathbf{z}}_{j-1} + \mathbf{H}_j^T \mathbf{R}_j^{-1} \mathbf{z}_j \right). \quad (3.11)$$

Substituting  $j = k$  in Equation (3.11) gives

$$\left( \mathbf{C}_{k,k-1}^T \mathbf{W}_{k,k-1} \mathbf{C}_{k,k-1} + \mathbf{H}_k^T \mathbf{R}_k^{-1} \mathbf{H}_k \right) \hat{\mathbf{x}}_{k|k} = \mathbf{C}_{k,k-1}^T \mathbf{W}_{k,k-1} \bar{\mathbf{z}}_{k-1} + \mathbf{H}_k^T \mathbf{R}_k^{-1} \mathbf{z}_k.$$

Substituting  $j = k - 1$  into Equation (3.8) and adding  $\mathbf{H}_k^T \mathbf{R}_k^{-1} \mathbf{H}_k \hat{\mathbf{x}}_{k|k-1}$  to both sides yields

$$\left( \mathbf{C}_{k,k-1}^T \mathbf{W}_{k,k-1} \mathbf{C}_{k,k-1} + \mathbf{H}_k^T \mathbf{R}_k^{-1} \mathbf{H}_k \right) \hat{\mathbf{x}}_{k|k-1} = \mathbf{C}_{k,k-1}^T \mathbf{W}_{k,k-1} \bar{\mathbf{z}}_{k-1} + \mathbf{H}_k^T \mathbf{R}_k^{-1} \hat{\mathbf{x}}_{k|k-1}.$$

Subtracting these two results yields

$$\left( \mathbf{C}_{k,k-1}^T \mathbf{W}_{k,k-1} \mathbf{C}_{k,k-1} + \mathbf{H}_k^T \mathbf{R}_k^{-1} \mathbf{H}_k \right) (\hat{\mathbf{x}}_{k|k} - \hat{\mathbf{x}}_{k|k-1}) = \mathbf{H}_k^T \mathbf{R}_k^{-1} (\mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_{k|k-1}).$$

Now, if we take the standard definition for the Kalman gain

$$\mathbf{K}_k = (\mathbf{C}_{k,k-1}^T \mathbf{W}_{k,k-1} \mathbf{C}_{k,k-1} + \mathbf{H}_k^T \mathbf{R}_k^{-1} \mathbf{H}_k)^{-1} \mathbf{H}_k^T \mathbf{R}_k^{-1} \quad (3.12)$$

we arrive at

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k (\mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_{k|k-1}), \quad (3.13)$$

the classic Kalman “prediction-correction” update equation. This equation expresses  $\hat{\mathbf{x}}_{k|k}$ , the estimate of the state at time  $k$  given data observations  $\mathbf{z}_0 \dots \mathbf{z}_k$ , recursively in terms of quantities known at time  $k$ :  $\hat{\mathbf{x}}_{k|k-1}$ , the state estimate given data observations  $\mathbf{z}_0 \dots \mathbf{z}_{k-1}$ ,  $\mathbf{K}_k$ , the Kalman gain at time  $k$ , and the current observation  $\mathbf{z}_k$ . With this recursive formulation, the entire observation history is summarized in these few matrices, and the memory requirements for Kalman filtering become feasible.

### 3.1.4.2 Time update

In order to use the update in Equation (3.13), we need an expression for  $\hat{\mathbf{x}}_{k|k-1}$  as a function of  $\hat{\mathbf{x}}_{k-1|k-1}$ . We note from the definition of  $\bar{\mathbf{e}}_{k,k-1}$  that  $\bar{\mathbf{e}}_{k,k-1} = \bar{\mathbf{e}}_{k-1,k-1} - \mathbf{C}_{k,k-1} \mathbf{G}_{k-1} \mathbf{w}_{k-1}$ , giving us

$$\begin{aligned} \mathbf{W}_{k,k-1}^{-1} &= \text{Var}(\bar{\mathbf{e}}_{k,k-1}) \\ &= \text{Var}(\bar{\mathbf{e}}_{k-1,k-1} - \mathbf{C}_{k,k-1} \mathbf{G}_{k-1} \mathbf{w}_{k-1}) \\ &= \text{Var}(\bar{\mathbf{e}}_{k-1,k-1}) + \text{Var}(\mathbf{C}_{k,k-1} \mathbf{G}_{k-1} \mathbf{w}_{k-1}) \\ &= \mathbf{W}_{k-1,k-1}^{-1} + \mathbf{C}_{k,k-1} \mathbf{G}_{k-1} \text{Var}(\mathbf{w}_{k-1}) \mathbf{G}_{k-1}^T \mathbf{C}_{k,k-1}^T \\ &= \mathbf{W}_{k-1,k-1}^{-1} + \mathbf{C}_{k,k-1} \Phi_{k-1,k} \mathbf{G}_{k-1} \mathbf{Q}_{k-1} \mathbf{G}_{k-1}^T \Phi_{k-1,k}^T \mathbf{C}_{k,k-1}^T. \end{aligned}$$

Some manipulation yields

$$\begin{aligned} \mathbf{W}_{k,k-1} &= \mathbf{W}_{k-1,k-1} - \mathbf{W}_{k-1,k-1} \mathbf{C}_{k-1,k-1} \Phi_{k-1,k} \mathbf{G}_{k-1} (\mathbf{Q}_{k-1}^{-1} + \\ &\quad \mathbf{G}_{k-1}^T \Phi_{k-1,k}^T \mathbf{C}_{k-1,k-1}^T \mathbf{W}_{k-1,k-1} \mathbf{C}_{k-1,k-1} \Phi_{k-1,k} \mathbf{G}_{k-1})^{-1} \mathbf{G}_{k-1}^T \Phi_{k-1,k}^T \mathbf{C}_{k-1,k-1}^T \mathbf{W}_{k-1,k-1}. \end{aligned}$$

By Equation (3.6) we have  $\mathbf{C}_{k,k-1} = \mathbf{C}_{k-1,k-1}\Phi_{k-1,k}$ , giving us

$$\begin{aligned} \mathbf{C}_{k,k-1}^T \mathbf{W}_{k,k-1} &= \Phi_{k-1,k}^T \left\{ \mathbf{I} - \mathbf{C}_{k-1,k-1}^T \mathbf{W}_{k-1,k-1} \mathbf{C}_{k-1,k-1} \Phi_{k-1,k} \mathbf{G}_{k-1} (\mathbf{Q}_{k-1}^{-1} + \right. \\ &\quad \left. \mathbf{G}_{k-1}^T \Phi_{k-1,k}^T \mathbf{C}_{k-1,k-1}^T \mathbf{W}_{k-1,k-1} \mathbf{C}_{k-1,k-1} \Phi_{k-1,k} \mathbf{G}_{k-1})^{-1} \mathbf{G}_{k-1}^T \Phi_{k-1,k}^T \right\} \mathbf{C}_{k-1,k-1}^T \mathbf{W}_{k-1,k-1}. \end{aligned}$$

Therefore,

$$\begin{aligned} (\mathbf{C}_{k,k-1}^T \mathbf{W}_{k,k-1} \mathbf{C}_{k,k-1}) \Phi_{k,k-1} (\mathbf{C}_{k-1,k-1}^T \mathbf{W}_{k-1,k-1} \mathbf{C}_{k-1,k-1})^{-1} \mathbf{C}_{k-1,k-1}^T \mathbf{W}_{k-1,k-1} \\ = \mathbf{H}_{k,k-1}^T \mathbf{W}_{k,k-1}. \end{aligned}$$

Left-multiplying by  $(\mathbf{C}_{k,k-1}^T \mathbf{W}_{k,k-1} \mathbf{C}_{k,k-1})^{-1}$  and substituting Equation (3.8) for  $j = k-1$  and  $j = k$ , this yields

$$\Phi_{k,k-1} \hat{\mathbf{x}}_{k-1|k-1} = \hat{\mathbf{x}}_{k|k-1},$$

which reduces to

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{A}_{k-1} \hat{\mathbf{x}}_{k-1|k-1},$$

by the definition of the transition matrix.

### 3.1.4.3 Kalman gain and covariance updates

Our final matrix to obtain a recursive solution for is  $\mathbf{K}_k$ , the Kalman gain matrix. From Equations (3.12), (3.9), and (3.10) we have

$$\mathbf{K}_k = (\mathbf{C}_{k,k}^T \mathbf{W}_{k,k} \mathbf{C}_{k,k})^{-1} \mathbf{H}_k^T \mathbf{R}_k^{-1}.$$

Defining  $\mathbf{P}_{k,j} = (\mathbf{C}_{k,j}^T \mathbf{W}_{k,j} \mathbf{C}_{k,j})^{-1}$  and applying Equations (3.9) and (3.10), we can see that

$$\begin{aligned} \mathbf{P}_{k,k}^{-1} &= (\mathbf{C}_{k,k}^T \mathbf{W}_{k,k} \mathbf{C}_{k,k}) \\ &= \begin{bmatrix} \mathbf{C}_{k,k-1}^T & \mathbf{H}_k^T \end{bmatrix} \begin{bmatrix} \mathbf{W}_{k,k-1} & 0 \\ 0 & \mathbf{R}_k^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{C}_{k,k-1} \\ \mathbf{H}_k \end{bmatrix} \\ &= \mathbf{C}_{k,k-1}^T \mathbf{W}_{k,k-1} \mathbf{C}_{k,k-1} + \mathbf{H}_k^T \mathbf{R}_k^{-1} \mathbf{H}_k \\ &= \mathbf{P}_{k,k-1}^{-1} + \mathbf{H}_k^T \mathbf{R}_k^{-1} \mathbf{H}_k. \end{aligned}$$

Using some facts from linear algebra (see [85]) about inversion of block matrices, it can be shown that

$$\mathbf{P}_{k,k} = \mathbf{P}_{k,k-1} - \mathbf{P}_{k,k-1} \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_{k,k-1} \mathbf{H}_k^T + \mathbf{R}_k)^{-1} \mathbf{H}_k \mathbf{P}_{k,k-1}.$$

These substitutions can be made in Equation (3.12), giving us

$$\mathbf{K}_k = \mathbf{P}_{k,k-1} \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_{k,k-1} \mathbf{H}_k^T + \mathbf{R}_k)^{-1},$$

giving us a simpler form for  $\mathbf{P}_{k,k}$ ,

$$\mathbf{P}_{k,k} = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_{k,k-1}.$$

Furthermore, we can show that

$$\mathbf{P}_{k,k-1} = \mathbf{A}_{k-1} \mathbf{P}_{k-1,k-1} \mathbf{A}_{k-1}^T + \mathbf{G}_{k-1} \mathbf{Q}_{k-1} \mathbf{G}_{k-1}^T.$$

It can be seen that the  $\mathbf{P}_{k,k-1}$  matrix is nothing but the covariance matrix of the error,

$$\mathbf{P}_{k,k-1} = \text{Var}(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k-1})$$

$$\mathbf{P}_{k,k} = \text{Var}(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k})$$

In particular, note that  $\mathbf{P}_{0,0} = \text{Var}(\mathbf{x}_0)$ .

### 3.1.4.4 Summary

Finally, summarizing the various equations we have derived in this section, we have the classic Kalman filtering process for the linear stochastic system with state-space description:

$$\left\{ \begin{array}{l} \mathbf{P}_{0,0} = \text{Var}(\mathbf{x}_0^s) \\ \mathbf{P}_{k,k-1} = \mathbf{A}_{k-1}\mathbf{P}_{k-1,k-1}\mathbf{A}_{k-1}^T + \mathbf{G}_{k-1}\mathbf{Q}_{k-1}\mathbf{G}_{k-1}^T \\ \mathbf{K}_k = \mathbf{P}_{k,k-1}\mathbf{H}_k^T(\mathbf{H}_k\mathbf{P}_{k,k-1}\mathbf{H}_k^T + \mathbf{R}_k)^{-1} \\ \mathbf{P}_{k,k} = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_{k,k-1} \\ \hat{\mathbf{x}}_{0|0}^s = E(\mathbf{x}_0^s) \\ \hat{\mathbf{x}}_{k|k-1}^s = \mathbf{A}_{k-1}\hat{\mathbf{x}}_{k-1|k-1}^s \\ \hat{\mathbf{x}}_{k|k}^s = \hat{\mathbf{x}}_{k|k-1}^s + \mathbf{K}_k(\mathbf{z}_k^s - \mathbf{H}_k\hat{\mathbf{x}}_{k|k-1}^s) \\ k = 1, 2, \dots \end{array} \right. \quad (3.14)$$

### 3.1.5 Combining deterministic and stochastic solutions

It can be shown [86] that the recursive solution for the deterministic solution to the system given in Equation (3.3) is given by

$$\left\{ \begin{array}{l} \mathbf{x}_{k|k-1}^d = \mathbf{B}_{k-1}\mathbf{u}_{k-1} \\ \mathbf{x}_{k|k}^d = \mathbf{x}_{k|k-1}^d \\ k = 1, 2, \dots \end{array} \right. \quad (3.15)$$

After unifying notation, it is easy to superimpose Equations (3.14) and (3.15) to arrive at the classic Kalman filtering process:

$$\left\{ \begin{array}{l} \mathbf{P}_{0,0} = \text{Var}(\mathbf{x}_0) \\ \mathbf{P}_{k,k-1} = \mathbf{A}_{k-1}\mathbf{P}_{k-1,k-1}\mathbf{A}_{k-1}^T + \mathbf{G}_{k-1}\mathbf{Q}_{k-1}\mathbf{G}_{k-1}^T \\ \mathbf{K}_k = \mathbf{P}_{k,k-1}\mathbf{H}_k^T(\mathbf{H}_k\mathbf{P}_{k,k-1}\mathbf{H}_k^T + \mathbf{R}_k)^{-1} \\ \mathbf{P}_{k,k} = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_{k,k-1} \\ \hat{\mathbf{x}}_{0|0} = E(\mathbf{x}_0) \\ \hat{\mathbf{x}}_{k|k-1} = \mathbf{A}_{k-1}\hat{\mathbf{x}}_{k-1|k-1} + \mathbf{B}_{k-1}\mathbf{u}_{k-1} \\ \hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\mathbf{z}_k - \mathbf{D}_k\mathbf{u}_k - \mathbf{H}_k\hat{\mathbf{x}}_{k|k-1}) \\ k = 1, 2, \dots \end{array} \right. \quad (3.16)$$

## 3.2 Extended Kalman Filter

Often, the systems we are interested in are not linear, and therefore, the derivation above does not capture their dynamics exactly. Exact update equations can be derived (see [87]), but they are intractable. Fortunately, the robustness of the standard Kalman filter is often enough to compensate for nonlinearities in the system. However, sometimes a more precise approximation to the nonlinear system dynamics is needed. In the extended Kalman filter (EKF), we take the first term of a Taylor series expansion of the nonlinear parts of the system about the current estimate of that term.

In this section, we consider nonlinear models of the form

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k) + \mathbf{G}'_k(\mathbf{x}_k)\mathbf{w}_k \quad (3.17)$$

$$\mathbf{z}_k = \mathbf{h}_k(\mathbf{x}_k) + \mathbf{v}_k, \quad (3.18)$$

where  $\mathbf{f}_k$  and  $\mathbf{h}_k$  are vector valued functions with ranges of dimension  $n$  and  $q$  respectively, and  $\mathbf{G}'_k$  is a matrix valued function of dimension  $n \times p$  (see Table 3.2 for a notational summary). We



**Table 3.2** Notation used in this section.

Symbol	Definition	Dimension	Notes
$\mathbf{x}_k$	state (at the $k$ th time instant)	$n \times 1$	$n \geq 1$
$\mathbf{f}_k$	system function	$n \times 1$	
$\mathbf{G}'_k$	system noise function	$n \times p$	$1 \leq p \leq n$
$\mathbf{w}_k$	white process noise	$p \times 1$	
$\mathbf{z}_k$	measurement	$q \times 1$	$1 \leq q \leq n$
$\mathbf{h}_k$	measurement function	$q \times 1$	
$\mathbf{v}_k$	white measurement noise	$q \times 1$	
$\mathbf{R}_k$	covariance matrix for $\mathbf{v}_k$	$q \times q$	
$\mathbf{A}_k$	partial of system function	$n \times n$	
$\mathbf{G}_k$	system noise matrices	$n \times p$	
$\mathbf{Q}_k$	covariance matrix for $\mathbf{w}_k$	$p \times p$	
$\mathbf{H}_k$	partial of measurement matrices	$q \times n$	
$\mathbf{y}_k$	derived measurement	$q \times 1$	
$\mathbf{u}_k$	derived control input	$p \times 1$	
$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_{k k}$	state estimate at time $k$ given data $\mathbf{z}_0 \dots \mathbf{z}_k$	$n \times 1$	
$\hat{\mathbf{x}}_{k k-1}$	state estimate at time $k$ given data $\mathbf{z}_0 \dots \mathbf{z}_{k-1}$	$n \times 1$	

make the usual assumptions with respect to the correlation of the noise and initial conditions:

$$\begin{aligned}
E(\mathbf{w}_k \mathbf{w}_l^T) &= \mathbf{Q}_k \delta_{kl} \\
E(\mathbf{v}_k \mathbf{v}_l^T) &= \mathbf{R}_k \delta_{kl} \\
E(\mathbf{w}_k \mathbf{v}_l^T) &= E(\mathbf{w}_k \mathbf{x}_0^T) = E(\mathbf{v}_k \mathbf{x}_0^T) = 0.
\end{aligned}$$

In this filter, we approximate the system using a linear model. We chose the initial estimate  $\hat{\mathbf{x}}_0 = E(\mathbf{x}_0)$  and predicted position  $\hat{\mathbf{x}}_{1|0} = \mathbf{f}_0(\hat{\mathbf{x}}_0)$  to be consistent with this linear model. We formulate linear approximations for  $\hat{\mathbf{x}} = \hat{\mathbf{x}}_{k|k}$  for each  $k$  using the (linearly approximated) predictions

$$\hat{\mathbf{x}}_{k+1|k} = \mathbf{f}_k(\hat{\mathbf{x}}_k) \quad (3.19)$$

and the linear state space description

$$\begin{aligned}
\mathbf{x}_{k+1} &= \mathbf{A}_k \mathbf{x}_k + \mathbf{u}_k + \mathbf{G}_k \mathbf{w}_k \\
\mathbf{y}_k &= \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k
\end{aligned} \quad (3.20)$$

where  $\mathbf{A}_k$ ,  $\mathbf{u}_k$ ,  $\mathbf{G}_k$ ,  $\mathbf{y}_k$ , and  $\mathbf{H}_k$  are to be determined on-line as follows. Consider the updates at time  $j$ , so that  $\hat{\mathbf{x}}_{j|j-1}$  has already been determined via Equation (3.19). Now consider the

linear Taylor approximation of  $\mathbf{f}_k$  at  $\hat{\mathbf{x}}_{k|k-1}$  and of  $\mathbf{h}_k$  at  $\hat{\mathbf{x}}_{k|k-1}$ :

$$\begin{aligned}\mathbf{f}_k(\hat{\mathbf{x}}_k) &\approx \mathbf{f}_k(\hat{\mathbf{x}}_k) + \frac{\partial \mathbf{f}_k}{\partial \mathbf{x}_k}(\hat{\mathbf{x}}_k) \\ \mathbf{h}_k(\hat{\mathbf{x}}_k) &\approx \mathbf{h}_k(\hat{\mathbf{x}}_k) + \frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k}(\hat{\mathbf{x}}_k),\end{aligned}$$

where  $\frac{\partial \mathbf{f}_k}{\partial \mathbf{x}_k}(\hat{\mathbf{x}}_k)$  and  $\frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k}(\hat{\mathbf{x}}_k)$  are the Jacobians of  $\mathbf{f}$  and  $h$ , respectively, evaluated at  $\hat{\mathbf{x}}_k$ . That is,

$$\left[ \frac{\partial \mathbf{h}}{\partial \mathbf{x}_k}(\mathbf{x}_k) \right] = \begin{bmatrix} \frac{\partial h_1}{\partial x_k^1}(\mathbf{x}_k) & \cdots & \frac{\partial h_1}{\partial x_k^n}(\mathbf{x}_k) \\ \vdots & \ddots & \vdots \\ \frac{\partial h_m}{\partial x_k^1}(\mathbf{x}_k) & \cdots & \frac{\partial h_m}{\partial x_k^n}(\mathbf{x}_k) \end{bmatrix} \quad (3.21)$$

and

$$\left[ \frac{\partial \mathbf{f}}{\partial \mathbf{x}_k}(\mathbf{x}_k) \right] = \begin{bmatrix} \frac{\partial f_1}{\partial x_k^1}(\mathbf{x}_k) & \cdots & \frac{\partial f_1}{\partial x_k^n}(\mathbf{x}_k) \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_k^1}(\mathbf{x}_k) & \cdots & \frac{\partial f_m}{\partial x_k^n}(\mathbf{x}_k) \end{bmatrix}.$$

We now make the following approximations based on this expansion

$$\begin{aligned}\mathbf{A}_k &\approx \frac{\partial \mathbf{f}_k}{\partial \mathbf{x}_k}(\hat{\mathbf{x}}_k) \\ \mathbf{H}_k &\approx \frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k}(\hat{\mathbf{x}}_k) \\ \mathbf{u}_k &\approx \mathbf{f}_k(\mathbf{x}_k) - \mathbf{A}_k \hat{\mathbf{x}}_k\end{aligned} \quad (3.22)$$

$$\begin{aligned}\mathbf{G}_k &\approx \mathbf{G}'_k(\hat{\mathbf{x}}_k) \\ \mathbf{y}_k &\approx \mathbf{z}_k - \mathbf{h}_k(\hat{\mathbf{x}}_{k|k-1}) + \mathbf{H}_k \hat{\mathbf{x}}_{k|k-1}.\end{aligned} \quad (3.23)$$

The nonlinear model in Equations (3.17) and (3.18) is approximated by the linear model in Equation (3.2) with the appropriate substitutions. Note that the approximations have to be updated at every iteration, and that the linearization is only assumed to be locally valid (leading to the subtraction of the predicted values in Equations (3.22) and (3.23)). Using these

approximations and the results from Section 3.1, we arrive at the EKF update equation:

$$\begin{aligned}
\hat{\mathbf{x}}_{k|k} &= \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\mathbf{y}_k - \mathbf{H}_k \hat{\mathbf{x}}_{k|k-1}) \\
&= \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k((\mathbf{z}_k - \mathbf{h}_k(\hat{\mathbf{x}}_{k|k-1}) + \mathbf{H}_k \hat{\mathbf{x}}_{k|k-1}) - \mathbf{H}_k \hat{\mathbf{x}}_{k|k-1}) \\
&= \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\mathbf{z}_k - \mathbf{h}_k(\hat{\mathbf{x}}_{k|k-1})).
\end{aligned} \tag{3.24}$$

Combining Equation (3.24) with the previously derived standard Kalman filter, we arrive at the following set of equations, generally collectively called the *extended Kalman filter* (EKF):

$$\mathbf{P}_{0,0} = \text{Var}(\mathbf{x}_0) \tag{3.25}$$

$$\hat{\mathbf{x}}_0 = E(\mathbf{x}_0) \tag{3.26}$$

$$\mathbf{P}_{k,k-1} = \left[ \frac{\partial \mathbf{f}_{k-1}}{\partial \mathbf{x}_{k-1}}(\hat{\mathbf{x}}_{k-1}) \right] \mathbf{P}_{k-1,k-1} \left[ \frac{\partial \mathbf{f}_{k-1}}{\partial \mathbf{x}_{k-1}}(\hat{\mathbf{x}}_{k-1}) \right]^T + \mathbf{G}_{k-1}(\hat{\mathbf{x}}_{k-1}) \mathbf{Q}_{k-1} \mathbf{G}_{k-1}^T(\hat{\mathbf{x}}_{k-1}) \tag{3.27}$$

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{f}_{k-1}(\hat{\mathbf{x}}_{k-1}) \tag{3.28}$$

$$\mathbf{K}_k = \mathbf{P}_{k,k-1} \left[ \frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k}(\hat{\mathbf{x}}_{k|k-1}) \right]^T \left[ \left[ \frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k}(\hat{\mathbf{x}}_{k|k-1}) \right] \mathbf{P}_{k,k-1} \left[ \frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k}(\hat{\mathbf{x}}_{k|k-1}) \right]^T + \mathbf{R}_k \right]^{-1} \tag{3.29}$$

$$\mathbf{P}_{k,k} = \left[ \mathbf{I} - \mathbf{K}_k \left[ \frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k}(\hat{\mathbf{x}}_{k|k-1}) \right] \right] \mathbf{P}_{k,k-1} \tag{3.30}$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\mathbf{z}_k - \mathbf{h}_k(\hat{\mathbf{x}}_{k|k-1})) \tag{3.31}$$

### 3.3 Alternate Form of Extended Kalman Filter

The Kalman filter equations given above can be algebraically manipulated into a variety of forms. In this section, we will follow a derivation from [88] of an equivalent form of the EKF with several numerical advantages in our situation. First, we introduce some notation to simplify the forthcoming derivation:

$$\mathbf{H} \triangleq \frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k}(\hat{\mathbf{x}}_{k|k-1}).$$

Recall the covariance update Equation (3.30) and Kalman gain Equation (3.29), repeated here for convenience:

$$\begin{aligned}\mathbf{K}_k &= \mathbf{P}_{k,k-1} \left[ \frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k}(\hat{\mathbf{x}}_{k|k-1}) \right]^T \left[ \left[ \frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k}(\hat{\mathbf{x}}_{k|k-1}) \right] \mathbf{P}_{k,k-1} \left[ \frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k}(\hat{\mathbf{x}}_{k|k-1}) \right]^T + \mathbf{R}_k \right]^{-1} \\ \mathbf{P}_{k,k} &= \left[ \mathbf{I} - \mathbf{K}_k \left[ \frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k}(\hat{\mathbf{x}}_{k|k-1}) \right] \right] \mathbf{P}_{k,k-1}.\end{aligned}$$

If Equation (3.29) is substituted into Equation (3.30), we have

$$\begin{aligned}\mathbf{P}_{k,k} &= [\mathbf{I} - \mathbf{K}_k \mathbf{H}] \mathbf{P}_{k,k-1} \\ &= [\mathbf{I} - \mathbf{P}_{k,k-1} \mathbf{H}^T (\mathbf{H} \mathbf{P}_{k,k-1} \mathbf{H}^T + \mathbf{R})^{-1} \mathbf{H}] \mathbf{P}_{k,k-1} \\ &= \mathbf{P}_{k,k-1} - \mathbf{P}_{k,k-1} \mathbf{H}^T (\mathbf{H} \mathbf{P}_{k,k-1} \mathbf{H}^T + \mathbf{R})^{-1} \mathbf{H} \mathbf{P}_{k,k-1}.\end{aligned}$$

**Proposition 4:** *If the inverses of  $\mathbf{P}_{k,k}$ ,  $\mathbf{P}_{k,k-1}$ , and  $\mathbf{R}$  exist,  $\mathbf{P}_{k,k}$  can be written as:*

$$\mathbf{P}_{k,k}^{-1} = (\mathbf{P}_{k,k-1})^{-1} + \mathbf{H}^T \mathbf{R}^{-1} \mathbf{H}. \quad (3.32)$$

**Proof:**

$$\begin{aligned}\mathbf{I} &= \mathbf{P}_{k,k} \mathbf{P}_{k,k}^{-1} = [\mathbf{P}_{k,k-1} - \mathbf{P}_{k,k-1} \mathbf{H}^T (\mathbf{H} \mathbf{P}_{k,k-1} \mathbf{H}^T + \mathbf{R})^{-1} \mathbf{H} \mathbf{P}_{k,k-1}] [(\mathbf{P}_{k,k-1})^{-1} + \mathbf{H}^T \mathbf{R}^{-1} \mathbf{H}] \\ &= \mathbf{P}_{k,k-1} (\mathbf{P}_{k,k-1})^{-1} + \mathbf{P}_{k,k-1} \mathbf{H}^T \mathbf{R}^{-1} \mathbf{H} \\ &\quad - \mathbf{P}_{k,k-1} \mathbf{H}^T (\mathbf{H} \mathbf{P}_{k,k-1} \mathbf{H}^T + \mathbf{R})^{-1} \mathbf{H} \mathbf{P}_{k,k-1} (\mathbf{P}_{k,k-1})^{-1} \\ &\quad - \mathbf{P}_{k,k-1} \mathbf{H}^T (\mathbf{H} \mathbf{P}_{k,k-1} \mathbf{H}^T + \mathbf{R})^{-1} \mathbf{H} \mathbf{P}_{k,k-1} \mathbf{H}^T \mathbf{R}^{-1} \mathbf{H} \\ &= \mathbf{I} - \mathbf{P}_{k,k-1} \mathbf{H}^T [-\mathbf{R}^{-1} + (\mathbf{H} \mathbf{P}_{k,k-1} \mathbf{H}^T + \mathbf{R})^{-1} + (\mathbf{H} \mathbf{P}_{k,k-1} \mathbf{H}^T + \mathbf{R})^{-1} \mathbf{H} \mathbf{P}_{k,k-1} \mathbf{H}^T \mathbf{R}^{-1}] \mathbf{H} \\ &= \mathbf{I} - \mathbf{P}_{k,k-1} \mathbf{H}^T [-\mathbf{R}^{-1} + (\mathbf{H} \mathbf{P}_{k,k-1} \mathbf{H}^T + \mathbf{R})^{-1} (\mathbf{I} + \mathbf{H} \mathbf{P}_{k,k-1} \mathbf{H}^T \mathbf{R}^{-1})] \mathbf{H} \\ &= \mathbf{I} - \mathbf{P}_{k,k-1} \mathbf{H}^T [-\mathbf{R}^{-1} + (\mathbf{H} \mathbf{P}_{k,k-1} \mathbf{H}^T + \mathbf{R})^{-1} (\mathbf{H} \mathbf{P}_{k,k-1} \mathbf{H}^T + \mathbf{R}) \mathbf{R}^{-1}] \mathbf{H} \\ &= \mathbf{I} - \mathbf{P}_{k,k-1} \mathbf{H}^T [-\mathbf{R}^{-1} + \mathbf{R}^{-1}] \mathbf{H} \\ &= \mathbf{I}\end{aligned}$$

□

We now wish to eliminate  $\mathbf{P}_{k,k}$  from the equation for the Kalman gain  $\mathbf{K}_k$ . Beginning from Equation (3.29), we have

$$\mathbf{K}_k = \mathbf{P}_{k,k-1} \mathbf{H}^T (\mathbf{H} \mathbf{P}_{k,k-1} \mathbf{H}^T + \mathbf{R})^{-1}.$$

We next insert  $\mathbf{P}_{k,k}\mathbf{P}_{k,k}^{-1}$  and  $\mathbf{R}^{-1}\mathbf{R}$  into the equation, which does not alter the gain:

$$\begin{aligned}\mathbf{K}_k &= (\mathbf{P}_{k,k}\mathbf{P}_{k,k}^{-1})\mathbf{P}_{k,k-1}\mathbf{H}^T(\mathbf{R}^{-1}\mathbf{R})[\mathbf{H}\mathbf{P}_{k,k-1}\mathbf{H}^T + \mathbf{R}]^{-1} \\ &= (\mathbf{P}_{k,k}\mathbf{P}_{k,k}^{-1})\mathbf{P}_{k,k-1}\mathbf{H}^T\mathbf{R}^{-1}[\mathbf{H}\mathbf{P}_{k,k-1}\mathbf{H}^T\mathbf{R}^{-1} + \mathbf{I}]^{-1}.\end{aligned}$$

Finally, we substitute Equation (3.32) into this equation:

$$\begin{aligned}\mathbf{K}_k &= \mathbf{P}_{k,k}[(\mathbf{P}_{k,k-1})^{-1} + \mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}]\mathbf{P}_{k,k-1}\mathbf{H}^T\mathbf{R}^{-1}[\mathbf{H}\mathbf{P}_{k,k-1}\mathbf{H}^T\mathbf{R}^{-1} + \mathbf{I}]^{-1} \\ &= \mathbf{P}_{k,k}[\mathbf{I} + \mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\mathbf{P}_{k,k-1}]\mathbf{H}^T\mathbf{R}^{-1}[\mathbf{H}\mathbf{P}_{k,k-1}\mathbf{H}^T\mathbf{R}^{-1} + \mathbf{I}]^{-1} \\ &= [\mathbf{P}_{k,k}\mathbf{H}^T\mathbf{R}^{-1} + \mathbf{P}_{k,k}\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\mathbf{P}_{k,k-1}\mathbf{H}^T\mathbf{R}^{-1}][\mathbf{H}\mathbf{P}_{k,k-1}\mathbf{H}^T\mathbf{R}^{-1} + \mathbf{I}]^{-1} \\ &= \mathbf{P}_{k,k}\mathbf{H}^T\mathbf{R}^{-1}[\mathbf{I} + \mathbf{H}\mathbf{P}_{k,k-1}\mathbf{H}^T\mathbf{R}^{-1}][\mathbf{H}\mathbf{P}_{k,k-1}\mathbf{H}^T\mathbf{R}^{-1} + \mathbf{I}]^{-1} \\ &= \mathbf{P}_{k,k}\mathbf{H}^T\mathbf{R}^{-1}.\end{aligned}$$

Summarizing, we have

$$\begin{aligned}\mathbf{P}_{k,k}^{-1} &= (\mathbf{P}_{k,k-1})^{-1} + \mathbf{H}^T\mathbf{R}^{-1}\mathbf{H} \\ \mathbf{K}_k &= \mathbf{P}_{k,k}\mathbf{H}^T\mathbf{R}^{-1},\end{aligned}$$

or, reinserting the previous notation and repeating the other EKF equations for convenience,

$$\mathbf{P}_{0,0} = \text{Var}(\mathbf{x}_0)$$

$$\hat{\mathbf{x}}_0 = E(\mathbf{x}_0)$$

$$\mathbf{P}_{k,k-1} = \left[ \frac{\partial \mathbf{f}_{k-1}}{\partial \mathbf{x}_{k-1}}(\hat{\mathbf{x}}_{k-1}) \right] \mathbf{P}_{k-1,k-1} \left[ \frac{\partial \mathbf{f}_{k-1}}{\partial \mathbf{x}_{k-1}}(\hat{\mathbf{x}}_{k-1}) \right]^T + \mathbf{G}_{k-1}(\hat{\mathbf{x}}_{k-1})\mathbf{Q}_{k-1}\mathbf{G}_{k-1}^T(\hat{\mathbf{x}}_{k-1})$$

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{f}_{k-1}(\hat{\mathbf{x}}_{k-1})$$

$$\mathbf{P}_{k,k}^{-1} = \mathbf{P}_{k,k-1}^{-1} + \left[ \frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k}(\hat{\mathbf{x}}_{k|k-1}) \right]^T \mathbf{R}_k^{-1} \left[ \frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k}(\hat{\mathbf{x}}_{k|k-1}) \right] \quad (3.33)$$

$$\mathbf{K}_k = \mathbf{P}_{k,k} \left[ \frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k}(\hat{\mathbf{x}}_{k|k-1}) \right]^T \mathbf{R}_k^{-1} \quad (3.34)$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\mathbf{z}_k - \mathbf{h}_k(\hat{\mathbf{x}}_{k|k-1}))$$

This alternate form has some advantages and disadvantages over the form presented in Section 3.2. Each step requires the computation of two  $n \times n$  matrix inversions, instead of the

one  $q \times q$  inversion required by Equation (3.29). If the order of the state vector becomes large, this could lead to computational problems. However, this form has several advantages for our situation, where  $n$  is smaller than  $q$ . In Chapter 7,  $n = 3$  or  $n = 6$  and  $q = 44$ , so we get a computational advantage. More importantly, this form of the Kalman filter is not sensitive to poor conditioning of the matrix  $\mathbf{HP}_{k,k}\mathbf{H}^T + \mathbf{R}$ , as often happens in our case. See Section 4.5.2 for discussion on the reasons why this occurs. Another advantage of this form is the ability to express infinite uncertainty in the initial covariance estimate ( $\mathbf{P}_{0,0} = \infty$ ).

### 3.4 An Intuitive Look at the Kalman Filtering Equations

In order to understand how a Kalman filter works, it is helpful to observe a simple example. This section presents a slightly modified version of Maybeck's example of calculating a position with a Kalman filter from [89]. We wish to estimate the 1D bearing of a ship at sea. At time  $t_1$ , a measurement of the bearing is taken. This measurement is denoted  $z_1$ . Due to human error and inaccuracies in the measuring device, this measurement is not precise. Assume that the precision can be determined to have a standard deviation of  $\sigma_{z_1}$ . Given this information, a conditional probability function for the bearing of the ship at time  $t_1$  can be established. A plot of the conditional probability density function (cpdf)  $f_{x(t_1)|z(t_1)}(x|z_1)$  as a function of  $x$  is shown in Figure 3.2. This plot illustrates the probability of being in any one direction, based upon the single measurement.

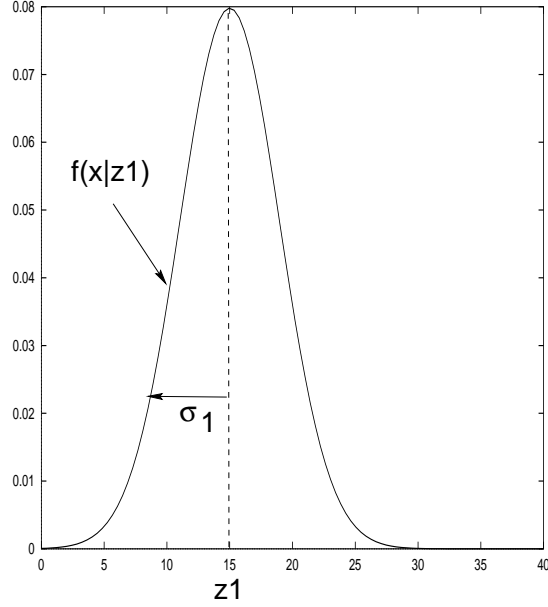
Based on this cpdf, the best estimate at time  $t_1$  for the bearing is

$$\hat{x}(t_1) = z_1$$

and the variance of the error in the estimate is

$$\sigma_x^2(t_1) = \sigma_{z_1}^2.$$

Note that  $\hat{x}$  is both the mode (peak) and the median (value with half of the probability weight to each side) of the cpdf, as well as the mean (center of mass).



**Figure 3.2** Conditional density of bearing based on measured value  $z_1$ .

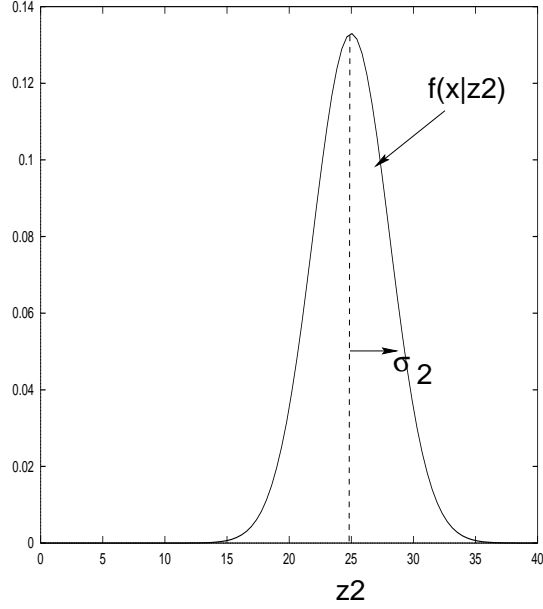
At time  $t_2$ , assume that a trained navigator takes an independent measurement of the ship's bearing, at a time  $t_2$ . Further assume that  $t_2 \approx t_1$ , so that the true bearing of the ship has not changed at all. This observation, like the first, has a measurement value  $z_2$  and a standard deviation  $\sigma_{z_2}$ . Because the expert has a higher skill and is more proficient with the equipment, assume that the uncertainty in the second measurement is lower than the uncertainty in the first. Figure 3.3 presents the cpdf for the bearing based on  $z_2$  alone. Note the narrow peak relative to Figure 3.2, indicating the relative certainty in the bearing based on this measurement.

At this point, there are two independent measurements of the bearing of the ship, with differing values and densities. The question is how to combine the data in some “optimal” fashion. It can be shown [89] that, based on the assumptions made, the cpdf for the bearing at time  $t_2 \approx t_1$ ,  $x(t_2)$ , given *both*  $z_1$  and  $z_2$ , is a Gaussian density with mean  $\mu$  and variance  $\sigma^2$ , with

$$\mu = \frac{\sigma_{z_2}^2}{\sigma_{z_1}^2 + \sigma_{z_2}^2} z_1 + \frac{\sigma_{z_1}^2}{\sigma_{z_1}^2 + \sigma_{z_2}^2} z_2 \quad (3.35)$$

$$1/\sigma^2 = 1/\sigma_{z_1}^2 + 1/\sigma_{z_2}^2 \quad (3.36)$$

as shown in Figure 3.4. Note that, from Equation (3.36),  $\sigma$  is less than either  $\sigma_{z_1}$  or  $\sigma_{z_2}$ , which



**Figure 3.3** Conditional density of bearing based on measured value  $z_2$ .

is to say that the uncertainty in the estimate of the bearing has been decreased by combining the two pieces of information.

Given this cpdf, the best estimate for the ship's bearing is

$$\hat{x}(t_2) = \mu \tag{3.37}$$

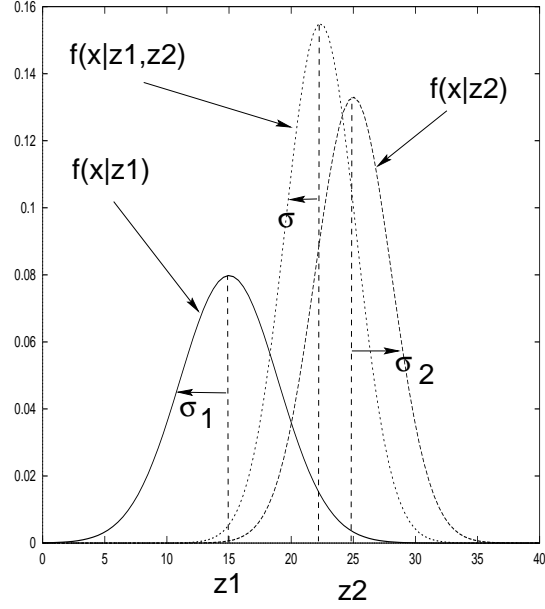
with an associated error variance

$$\sigma_x^2(t_2) = \sigma^2. \tag{3.38}$$

It is the mode and mean of the cpdf. Since it is based on a *conditional* pdf, it can also be referred to as the conditional mean. Some thought (see [89]) reveals that  $\mu$  is maximum likelihood estimate (MLE), the weighted least squares estimate, and the linear estimate whose variance is less than that of any other linear unbiased estimate. In other words, this is the “optimal” estimate for the bearing under almost any reasonable definition of optimality.

The form of  $\mu$  given in Equation (3.35) is intuitively pleasing. If the measurements  $z_1$  and  $z_2$  were of equal quality,  $\sigma_{z_1} = \sigma_{z_2}$ , the optimal estimate of the bearing is simply the average of the two measurements, as would be expected. The uncertainty in the new estimate is exactly





**Figure 3.4** Conditional density of bearing based on measured values  $z_1$  and  $z_2$ .

half of the uncertainty in each measurement. If the measurements are of unequal quality, say  $\sigma_{z_1} > \sigma_{z_2}$ , then Equation (3.35) dictates “weighting”  $z_2$  more heavily than  $z_1$ . In the extreme case, measurement  $z_1$  is much more uncertain than measurement  $z_2$ ,  $\sigma_{z_1} \gg \sigma_{z_2}$ , and the best combined estimate is approximately  $z_2$ ,  $\mu \approx z_2$ . Then

$$\sigma^2 = \frac{\sigma_{z_2}^2}{\sigma_{z_1}^2 + \sigma_{z_2}^2} \sigma_{z_1}^2 \approx \sigma_{z_2}^2.$$

Note that even in this case, the variance of the estimate is less than the variance of either measurement: even poor quality data provide some information and thus increase the precision of the filter output.

In order to better show the equivalence of this presentation with the Kalman filter presented earlier in this chapter, we can rewrite Equation (3.37) as

$$\hat{x}(t_2) = \mu \tag{3.39}$$

$$= \frac{\sigma_{z_2}^2}{\sigma_{z_1}^2 + \sigma_{z_2}^2} z_1 + \frac{\sigma_{z_1}^2}{\sigma_{z_1}^2 + \sigma_{z_2}^2} z_2 \tag{3.40}$$

$$= \left(1 - \frac{\sigma_{z_1}^2}{\sigma_{z_1}^2 + \sigma_{z_2}^2}\right) z_1 + \frac{\sigma_{z_1}^2}{\sigma_{z_1}^2 + \sigma_{z_2}^2} z_2 \tag{3.41}$$

$$= z_1 + \frac{\sigma_{z_1}^2}{\sigma_{z_1}^2 + \sigma_{z_2}^2} (z_2 - z_1) \tag{3.42}$$

or in the form actually used in the Kalman filter implementations (noting that  $\hat{x}(t_1) = z_1$ ),

$$\hat{x}(t_2) = \hat{x}(t_1) + K(t_2)[z_2 - \hat{x}(t_1)] \tag{3.43}$$

$$K(t_2) = \frac{\sigma_{z_1}^2}{\sigma_{z_1}^2 + \sigma_{z_2}^2}. \tag{3.44}$$

These equations show that the optimal estimate at time  $t_2$ ,  $\hat{x}(t_2)$ , is equal to the best prediction of its value before measurement  $z_2$  is taken,  $\hat{x}(t_1)$ , plus a correction term of an optimal weighting value times the difference between  $z_2$  and the best prediction of its value just before  $z_2$  is taken,  $\hat{x}(t_1)$ . This “predictor-corrector” structure should be familiar from Equations (3.13) and (3.31). Based on all previous information, a prediction of the desired variables and the measurement (a function of those variables) is made. Then when the next measurement is taken, the difference between it and its predicted value is used to “correct” the prediction of the desired variables.

Using the  $K(t_2)$  form given in Equation (3.44), Equation (3.36) can be rewritten as

$$\sigma_x^2(t_2) = \sigma^2 \tag{3.45}$$

$$= \frac{1}{1/\sigma_{z_1}^2 + 1/\sigma_{z_2}^2} \tag{3.46}$$

$$= \frac{\sigma_{z_1}^2 \sigma_{z_2}^2}{\sigma_{z_1}^2 + \sigma_{z_2}^2} \tag{3.47}$$

$$= \frac{\sigma_{z_2}^2}{\sigma_{z_1}^2 + \sigma_{z_2}^2} \sigma_{z_1}^2 \tag{3.48}$$

$$= \left(1 - \frac{\sigma_{z_1}^2}{\sigma_{z_1}^2 + \sigma_{z_2}^2}\right) \sigma_{z_1}^2 \tag{3.49}$$

$$= [1 - K(t_2)] \sigma_{z_1}^2 \tag{3.50}$$

$$= \sigma^2(t_1) - K(t_2) \sigma^2(t_1) \tag{3.51}$$

recalling that  $\sigma_x^2(t_1) = \sigma_{z_1}$  and  $\sigma_x^2(t_2) = \sigma$ . Note that since we have assumed that the cpdf is Gaussian, the values of  $\hat{x}(t_2)$  and  $\sigma_x^2(t_2)$  embody all information in  $f_{x(t_2)|z(t_1),z(t_2)}(x|z_1, z_2)$ . In other words, by propagating these two variables, the conditional density of the bearing at time  $t_2$ , given  $z_1$  and  $z_2$ , is completely specified.

This example illustrates the power of the Kalman filtering algorithm. It uses information about the uncertainty of the current estimate for a variable and information about the uncertainty of a measurement to arrive at the best *combination* of these quantities to use as a new estimate for the variable. This example does not include all the features of the Kalman filter, however. The dynamics of the problem, or the movement of the state that occurs when  $t_2 \neq t_1$ , are not considered here. Maybeck [89] continues this example to include a dynamic model for the system and shows the continued parallel between the Kalman filtering equations and the “intuitive” answer arrived at by considering the Gaussian cpdfs directly.

### 3.5 Robustness and Limitations of Kalman Filtering

The extended Kalman filter (EKF) is a very robust and flexible algorithm. This fact is evidenced by the wide use of the EKF, although many of the assumptions involved in the

derivation of the filter are violated in most common situations. However, Kalman filter design is not always a straightforward process. The noise covariance matrices  $Q$  and  $R$  are often known only to within an order of magnitude. The process and observation noise processes, being real, cannot be white. In spite of all these violations of the theoretical assumptions, the EKF gives reasonable results in a wide array of applications, including many applications very similar to ours. Bierman [90] outlines some practical problems involved with Kalman filter design.

Least squares techniques in general, and Kalman filtering in particular, have some well-known limitations when applied to parameter estimation. The assumptions made by these techniques exclude gross errors (outliers), systematic errors and correlations in observations, and assume exact models [91]. Each of these assumptions is often violated to a certain extent in any given observation; these violations may prevent least-squares techniques from yielding acceptable results. Estimators have been developed to reduce or eliminate problems stemming from these types of errors [92], but the flexibility and overall robustness of the EKF, particularly when combined with a simple outlier reduction test (see [91]), led us to use the EKF in favor of the more numerically robust methods [92].

## CHAPTER 4

### A MODEL-BASED OBJECT TRACKING SYSTEM

In this chapter, a system for model-based object tracking is described. This system updates the internal parameters of an object model from monocular grayscale images of that object. The system accounts for both self-occlusion and external occlusion of features, and weights feature observations according to their predicted usefulness in disambiguating the state, as well as the amount of spatial uncertainty present in the feature observation.

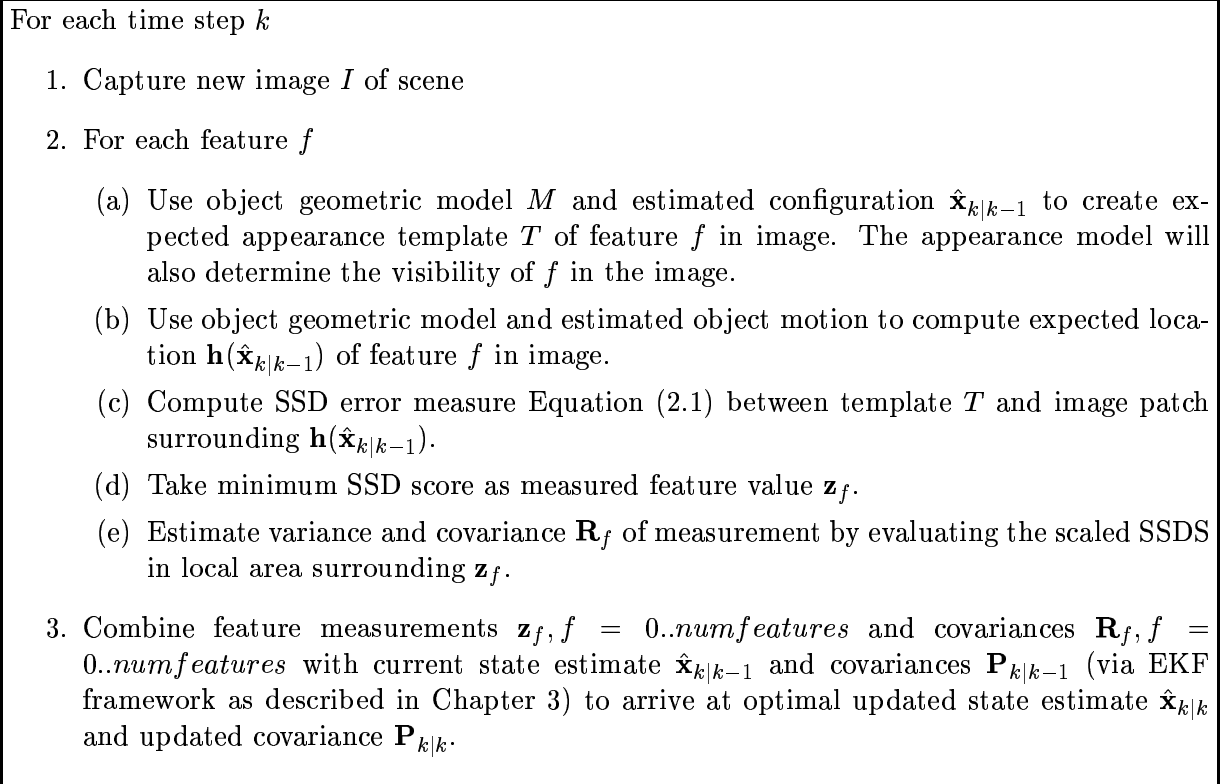
We begin by presenting the tracking algorithm used in the system. Then we examine several components used in the tracking algorithm: feature tracking and measurement uncertainty estimation, assimilation of feature tracking results, and finally provisions for self-occlusion and external occlusion of features.

#### 4.1 Algorithm

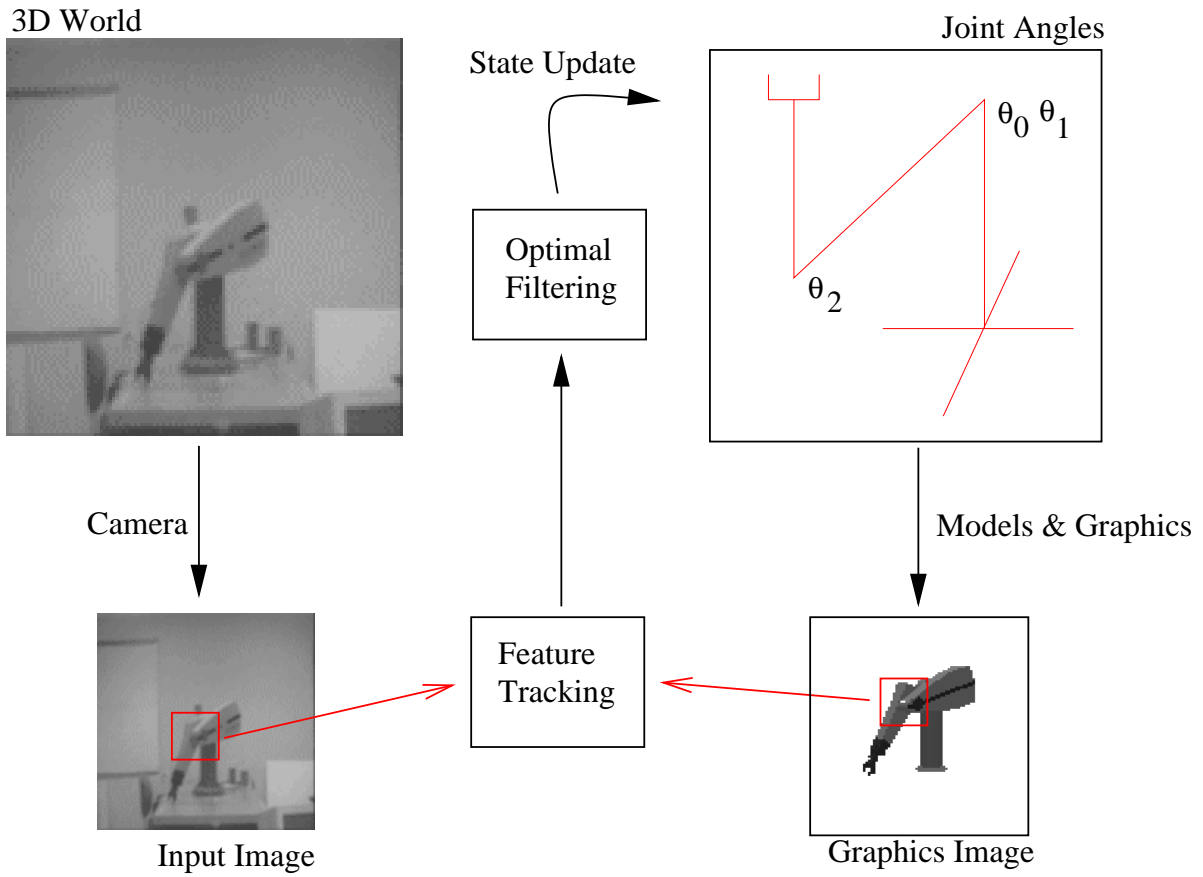
In this section, we present an overview of our algorithm for object tracking. Figure 4.1 shows the pseudocode for this algorithm, and Figure 4.2 illustrates the process pictorially.

#### 4.2 Feature Tracking

In this section, we describe the portion of the algorithm involved in generating the feature tracking results described above. First, a description of the models used and assumptions made will be presented. Then, a description of the generation of synthetic scenes will be presented.



**Figure 4.1** TRACK-OBJECT, an algorithm for object tracking.



**Figure 4.2** Tracking system overview.

The use of this scene for generating feature appearance templates will be discussed. Finally, the a dissimilarity metric for comparing a feature appearance template to a portion of the input image, and its use in feature tracking, will be presented.

### 4.2.1 Models used

In this thesis, five models will be used: the overall system model, the object geometric model, the object appearance model, the imaging model, and the object dynamic model. In this section, we will describe these models and discuss the assumptions that can be made for each.

#### 4.2.1.1 System model

We utilize a general nonlinear model in our tracking system. The use of the system model in tracking was described in Chapter 3, and examples for some different situations will be presented in Chapter 6. The other models we will describe relate to assumptions about object geometry, appearance, and dynamics, as well as the the imaging system, and are realized by instantiating functions in this system model. In this model, the *state vector* at time  $k + 1$ ,  $\mathbf{x}_{k+1}$ , is given by a vector valued function  $\mathbf{f}_k$  of the state vector at time  $k$ ,  $\mathbf{x}_k$ , with additive noise. Observations of the system are given by another vector valued function  $\mathbf{h}_k$  of the state vector at time  $k$ ,  $\mathbf{x}_k$ , with additive noise. The equations for the nonlinear model used for the system are of the form

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k) + \mathbf{w}_k$$

$$\mathbf{z}_k = \mathbf{h}_k(\mathbf{x}_k) + \mathbf{v}_k.$$



See Table 3.2 for a notational summary. We make the usual (in the Kalman filtering literature) assumptions with respect to the correlation of the noise and initial conditions:

$$\begin{aligned}
 E(\mathbf{w}_k \mathbf{w}_l^T) &= \mathbf{Q}_k \delta_{kl} \\
 E(\mathbf{v}_k \mathbf{v}_l^T) &= \mathbf{R}_k \delta_{kl} \\
 E(\mathbf{w}_k \mathbf{v}_l^T) &= E(\mathbf{w}_k \mathbf{x}_0^T) = E(\mathbf{v}_k \mathbf{x}_0^T) = 0.
 \end{aligned}$$

#### 4.2.1.2 Object geometric model

The object geometric model describes the size and shape of each link of the articulated object under consideration and how the links move with respect to one another. This model is a superset of the standard kinematic model used in robotics, which defines only the interrelation of the coordinate systems of each link, but not the shapes of the links [93]. Thus, the object geometric model will determine the position of all features in the world coordinate frame, given the robot's configuration parameters. The use of complex explicit geometric models in object tracking has increased in recent years [94], [95], [96]. We assume that the object model is known *a priori*, and that it does not change.

#### 4.2.1.3 Object appearance model

The object appearance model describes the color, texture, and materials used on each link described by the object geometric model. The combination of the object appearance model and the object geometric model defines the position and appearance of every point on the object. Specifically, this allows us to recover the 3D appearance of the area immediately surrounding a feature, given the feature location and the object configuration [11]. The method used to do this is described in Chapter 5.

In addition to direct modeling of object attributes, this model includes assumptions such as ambient lighting and background color that are not strictly object features, but that will affect the appearance of features.

#### 4.2.1.4 Imaging model

The imaging model mathematically describes the camera used to image the scene. This is a superset of the perspective or orthogonal projection models, including any lens modeling (such as vignetting, lens distortion, or focusing effects) for the camera [23]. We include the position of the camera in the scene in this model. A fixed camera position could also be assumed, by including the object position relative to the camera in the object geometric model. This model is assumed to be known and constant.

#### 4.2.1.5 Object dynamic model

The dynamic model describes the assumptions about motion through the filter’s *state space*. In our case,  $\mathbf{x}_k$  contains the joint angles  $\mathbf{q}$  and joint velocities  $\dot{\mathbf{q}}$ , so the object dynamic model describes movement through the robot’s configuration space. For an in-depth description of configuration space concepts, see [93] or [97]. Intuitively, the dynamic model  $\mathbf{f}_k(\mathbf{x}_k)$  is the best estimate (from a modeling standpoint) of the next state of the system given the current state  $\mathbf{x}_k$ . The choice of these models was described in Section 2.4.

For instance, if we know that the robot is being controlled such that it maintains a constant velocity in configuration space, we adopt a “constant velocity” dynamic model

$$\mathbf{f}_k(\mathbf{x}_k) = \mathbf{q}_k + \Delta \dot{\mathbf{q}}_k,$$

where  $\Delta$  is the sampling interval. If we wish to avoid explicit modeling of motion, we can assume a “constant position” dynamic model:

$$\mathbf{f}_k(\mathbf{x}_k) = \mathbf{x}_k.$$

Note that since  $\mathbf{x}_{k+1} = \mathbf{f}_k(\mathbf{x}_k) + \mathbf{w}_k$ , this is not the same as assuming that the object never moves, but rather that the motion is completely unpredictable. Specifically, deviations from the motion model are assumed to be zero-mean, Gaussian, and white. If “no dynamic model” were assumed, the implicit assumption would be what we have described as a constant position

dynamic model. In this case, the state vector is simply the object configuration parameters,

$$\mathbf{x}_k = \mathbf{q}_k.$$

If constant velocity is assumed, the velocity through configuration space needs to be estimated as well, so

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{q}_k \\ \dot{\mathbf{q}}_k \end{bmatrix}.$$

Motion models  $\mathbf{f}$  could also be formulated to model constant workspace end-effector velocities or a number of other feasible dynamic models.

#### 4.2.1.6 Errors in models

Each of these models affects the robustness of the visual tracking process, and errors in each model will degrade the performance in different ways. Errors in the appearance and imaging models will affect the appearance of a feature template and will degrade the feature-tracking portion of the algorithm in as much as the appearance of the features in the input images no longer match the expected appearance. As long as the “correct” location of a feature is most similar to the template, feature tracking will occur without errors. Since the correlation measure does not account for changes in illumination [50], errors in the modeling of illumination would change the object appearance, and would affect feature tracking in much the same way.

Errors in the object geometric model would affect the system differently. First, the forward kinematics of the object describe the mapping between the object configuration space and the workspace. Thus, errors in this mapping will initialize the feature trackers at incorrect positions in the image, positions different from the actual locations of the features. Second, the forward kinematics as well as the partial derivatives of this map are used to compute a linearization of this nonlinear map for the purpose of assimilating feature tracking results into an updated object state. Section 3.2 describes this process. The implication of an incorrect geometric model on the process is that a minimum least squares fit to the object feature locations is computed,

and the estimates of the object parameters would be incorrect in so far as the placement of the features is incorrect with respect to the object due to the geometric errors. Luckily, since the same geometric map is used for the forward projection of feature locations and the assimilation of feature tracking results, small errors in the geometric map result in proper feature tracking at the expense of accurate object configuration estimates. If feature location prediction and object dynamic models were used to a greater extent, such that only the most likely feature locations in the image were searched, errors in the geometric map might be more problematic.









Errors in the dynamic model will draw the state estimate off during the time projection portion of the object-tracking system. This means that the predicted feature locations will be incorrect to a certain extent. As long as the actual feature locations still fall within the search regions, the only result is that the correction portion of the prediction-correction update equation (see Section 3.2 for an explanation of the extended Kalman filter) will be larger. If too many features fall outside the search region due to an incorrect dynamic model, the feature trackers will not be able to bring the full state estimate close enough to the actual state, and tracking will fail.

#### 4.2.2 Scene rendering

Given an object geometric model, an object appearance model, an imaging model, and an estimated configuration of the object, a *synthetic scene* can be generated. This scene represents the system's best estimate of the appearance of the scene, as imaged by the camera.

A 3D graphics object representing each link of the object is generated using OpenGL [98] graphics primitives. These objects, for the case of the PUMA arm, are shown in Figure 4.3. These models allow the user to view the links from any specified viewpoint.

OpenGL allows the use of homogeneous transformation matrices to move between the world coordinate frame and subordinate model coordinate frames, much as homogeneous transformation matrices are used in robotics to move between the world coordinate frame and subordinate

	 <p>Full Arm</p>	
 <p>Link 0</p>	 <p>Link 1</p>	 <p>Link 2</p>
 <p>Link 3</p>	 <p>Link 4</p>	 <p>Link 5</p>
	 <p>Link 6</p>	

**Figure 4.3** Individual link graphics objects.

coordinate frames rigidly attached to the links of the robot [93]. The coordinate frames for each of the six links for the PUMA are illustrated in Figure 4.4.

The object geometric model and configuration completely specify the relative locations of the link coordinate frames with respect to the world coordinate frame. Thus, the 3D graphics objects representing the links can be placed in the world coordinate frame of the graphics scene, in the locations that they would occupy when the arm is in a particular configuration. In order to determine the appearance of the scene, an imaging model such as described above is assumed. With these models in place a synthetic scene such as that shown in Figure 4.5 can be generated. Note that the image produced by the program uses 24 bits of grayscale and currently is produced at a resolution of  $512 \times 485$ . Therefore, the quality of the image in this thesis does not reflect the quality of the synthetic scene actually used by the system.

### 4.2.3 Generating feature appearance templates

Feature points are specified by hand as 3D points in a link coordinate system. Given the geometric model and an estimated configuration, the position of each feature point in the world coordinate frame can be determined. Given the imaging model, the projection of each feature point onto the image plane can be determined. We term this point in the image plane the *expected feature location*.

If a feature point is visible in a given configuration, it can be expected that the area in the input image surrounding that feature point will resemble the area in the synthetic image surrounding the expected feature location. This is the principle upon which we base our method for automatically generating templates for arbitrary complex features. A fixed rectangular area of the synthetic image called the *feature appearance template*, centered on the expected feature location, is saved for use during feature tracking. Some other choices for feature templates are discussed in Section 2.5.3.1. This choice for template generation enables us to use arbitrary points on the surface of the object as features, and allows the use of the same framework whether

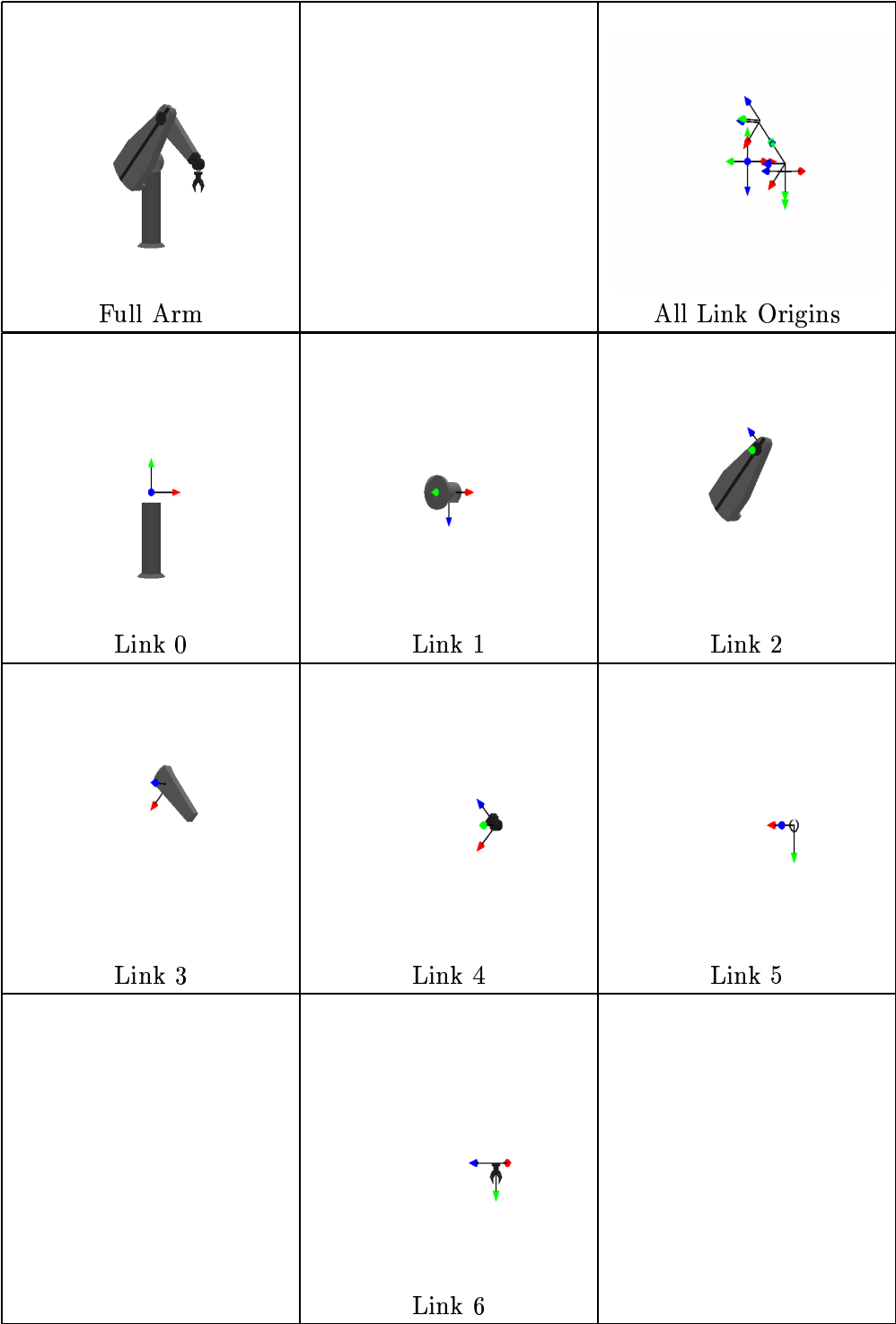
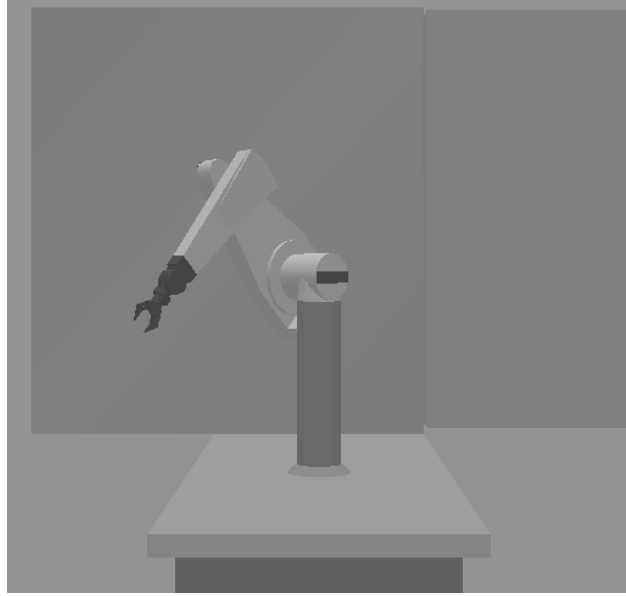


Figure 4.4 Link coordinate frames.



**Figure 4.5** A synthetic image.

the feature is a line, a spot, a corner, or the center of the letter P. This process is repeated for each visible feature.

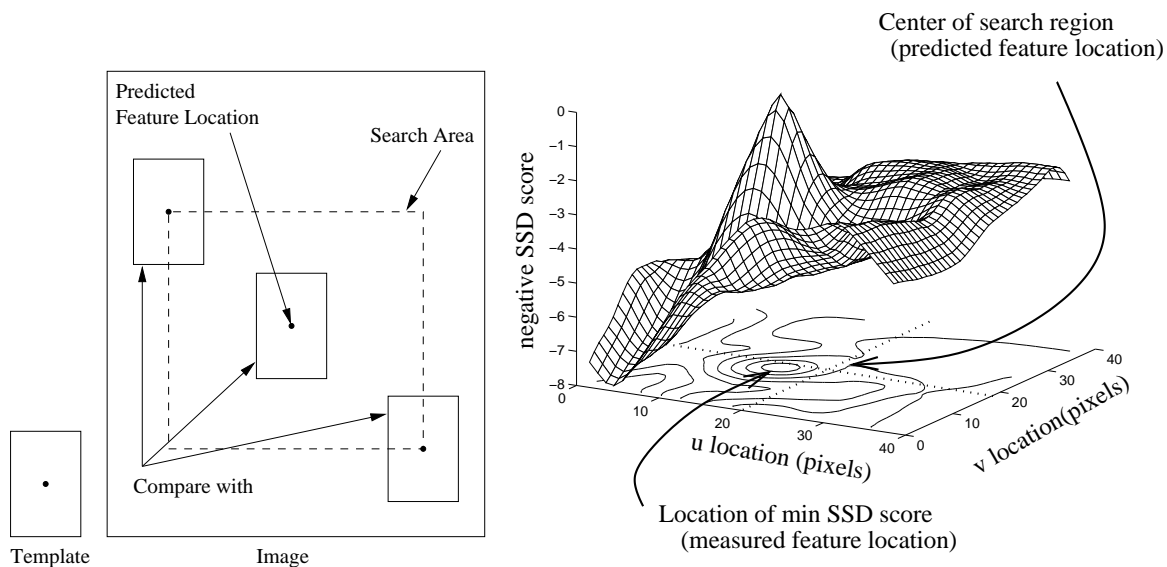
A feature is defined as visible if the projection ray from the feature location to the optical center of the camera does not intersect any portion of the object in the estimated configuration.

#### **4.2.4 Use of SSD in feature tracking**

In this section, we investigate the use of the SSD measure Equation (2.1) for feature extraction. We will describe the information about a feature that can be extracted from the SSD surface, and how to extract that information.

We define the *SSD surface* (SSDS) to be that surface generated by the SSD match score of a template at each pixel in some region of the image. The full SSD surface (where the image region under consideration is equal to the entire image) will be the same size as the image, ignoring border effects. It is expected that if the template matches well with a subimage, the SSD match score at that location will be low, and the SSD surface will exhibit a minimum there.



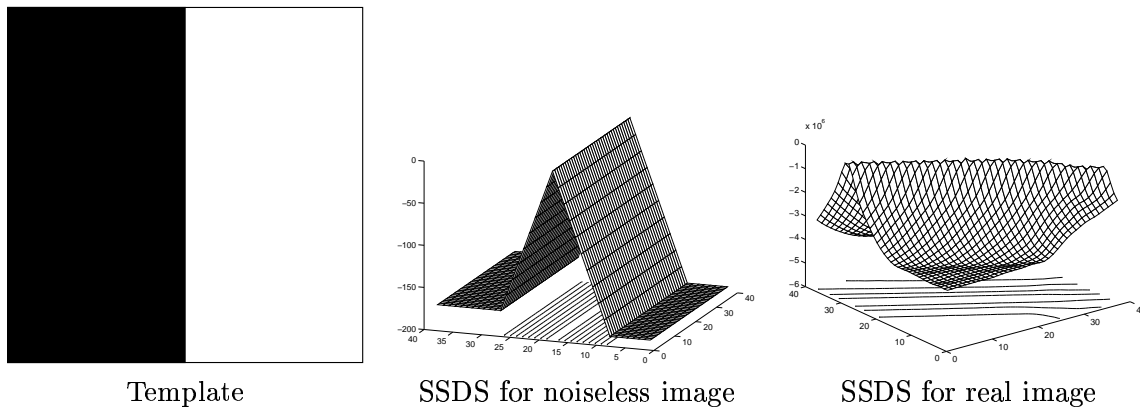


**Figure 4.6** Use of similarity metric.

The SSD surface is used for feature detection in an image in the following way. Section 2.5.3.3 describes the general theory of the use of dissimilarity measures for feature tracking. The template is initialized to the expected appearance of the feature of interest, as described in Section 4.2.3. The SSD surface is computed in some neighborhood of the expected feature location in the image as given by Equation (2.1) and the location of the lowest SSD match score is returned as the most likely location of the feature. This process is pictorially illustrated in Figure 4.6.

#### 4.2.5 Spatial discrimination

As the SSD measure is used to compare the template to areas of the image near the minimum, some measure of the *spatial discrimination* power of the template can be generated [15] [12]. Spatial discrimination is defined as the ability to detect feature motion along a given direction in the image. This concept is quite similar to the confidence measures discussed in Section 2.5.3.4 that estimate the reliability of the location estimate, but we interpret the confidences as spatial uncertainties in the returned location. For example, an *edge feature*, such

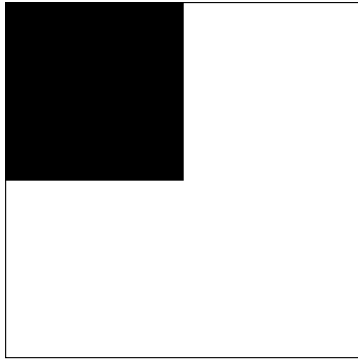


**Figure 4.7** An edge feature.

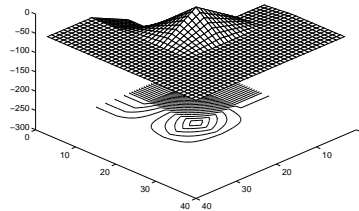
as that shown in Figure 4.7, can detect feature movement only in the direction orthogonal to the edge. In this example, horizontal movement of the edge in the image can be tracked, but vertical movement cannot be tracked. This fact can be determined by examining the SSDS shown in Figure 4.7. Note that the negative of the SSDS is shown for illustrative purposes. The surface has a sharp minimum at the edge, in the direction perpendicular to the edge. In the direction parallel to the edge, the match scores are approximately equal, forming a surface with little curvature in those directions.

Figure 4.8 shows an SSDS that corresponds to a corner feature, and that has a sharp minimum at the extremal point. This shows that, in contrast to the edge feature described above, this surface has sharp curvature in all directions around the minimum, and provides good localization in both dimensions. Indeed, for a point feature such as a corner the curvature in each direction is approximately equal.

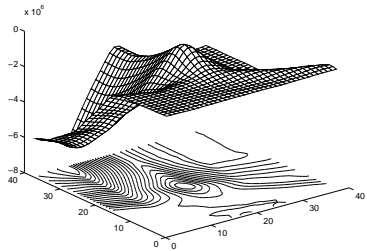
Figure 4.9 shows an SSDS that corresponds to a point belonging to a homogeneous area, and that exhibits no sharp minima. In the ideal case, the SSDS is flat. In real images, multiple small local minima in all directions will exist. In all directions, the match scores are approximately equal, forming a surface with little curvature in any direction. This surface does not discriminate the spatial location of the template well in any direction.



Template



SSDS for noiseless image

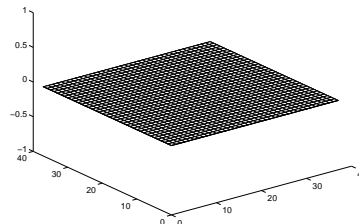


SSDS for real image

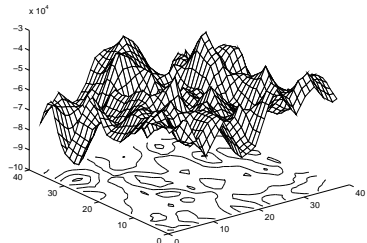
**Figure 4.8** A corner feature.



Template



SSDS for noiseless image



SSDS for real image

**Figure 4.9** A homogeneous feature.

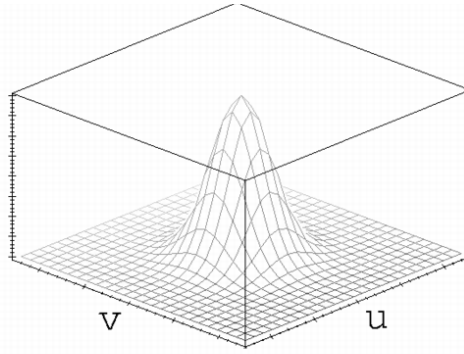
While conclusions about the efficacy of a given template for feature localization can be drawn from the fully computed SSDS, it is expensive both computationally and from a computer memory standpoint to maintain the complete surface for this purpose. In the next section, we use a normalized version of the SSDS to compute the probability density function of a Gaussian random vector, and illustrate that the vector retains approximately the same information about spatial feature discrimination as the full SSDS.

#### 4.2.6 Approximation for $\mathcal{RD}$

In order to maintain and use relevant information about the shape of the response distribution, we introduce a mathematical approximation to the distribution given in Equation (2.2). By suppressing the off-peak response of the feature tracking result, this response distribution function converts the SSDS into an approximately Gaussian distribution that contains the feature tracking information we wish to maintain in the Kalman filter.

We model the feature location as a 2D random vector and show that the mean and covariance of this vector contain relevant information about the ability of the template in localizing a feature. The extended Kalman filter described in Section 3.2 models measurements as Gaussian random vectors. This approximation will allow us to explicitly recognize the assumptions made in a Kalman filtering framework, by explicitly analyzing the density functions of the measurements used in the filter.

In Section 4.2.6.1 we review the characteristics of Gaussian random vectors and illustrate the behavior of the density functions of these vectors in several situations. In Section 4.2.6.2 we describe a method for estimating the parameters for a Gaussian random vector given a feature measurement as described above.



**Figure 4.10** Density function for a 2D Gaussian random vector.

#### 4.2.6.1 Gaussian random vector

A 2D Gaussian random vector has a density function as shown in Figure 4.10. The equation for this function is

$$f_{\mathbf{x}}(\mathbf{x}) = \frac{1}{2\pi\sqrt{\det(\mathbf{R})}} \times \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \mathbf{R}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right],$$

where  $\mathbf{x} \triangleq [u \ v]^T$ ,  $\boldsymbol{\mu} \triangleq [E(u) \ E(v)]^T$ , and  $\mathbf{R} \triangleq \begin{bmatrix} \text{Var}^2(u) & \text{Cov}(u, v) \\ \text{Cov}(u, v) & \text{Var}^2(v) \end{bmatrix}$ .

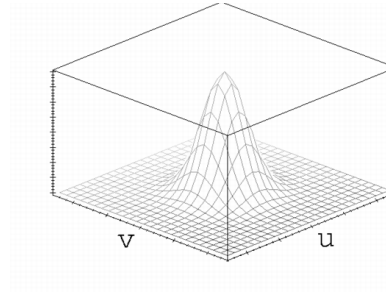
It has been noted [89], [99] that the covariance matrix of a random vector determines the shape and orientation of the contours (in the noiseless case, they are ellipsoids) of constant probability. Kosaka and Kak [39] provide an in-depth discussion on the equivalence between a covariance matrix and the related error ellipsoids. Figures 4.11, 4.12, 4.13, and 4.14 show the covariance matrices, density functions, and contours of constant probability for several different situations. The orientations of the semimajor axes of the error ellipsoids determine the eigenvectors of the covariance matrix. The lengths of the semimajor axes determine the eigenvalues of the covariance matrix.

If the measurement vector  $\mathbf{z}_k$  is interpreted as an uncertain location in the  $(u, v)$  plane, it is illustrative to analyze the behavior of the density function as  $\mathbf{R}_k$  changes. For example, if  $\mathbf{R}_k = \sigma^2 I$ , the location is equally certain in each direction, as shown in Figure 4.11. If

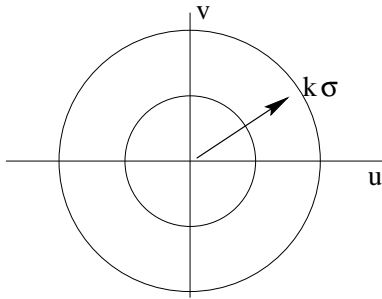
Covariance Matrix

$$\mathbf{R}_k = \begin{bmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{bmatrix}$$

Density Function



Contours of Constant Probability



Notes

$$\sigma = \sigma_u = \sigma_v$$

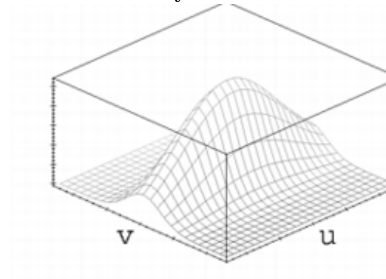
$$\rho_{uv} = 0$$

**Figure 4.11** Point feature characteristics.

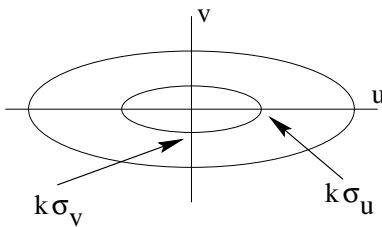
Covariance Matrix

$$\mathbf{R}_k = \begin{bmatrix} \sigma_u^2 & 0 \\ 0 & \sigma_v^2 \end{bmatrix}$$

Density Function



Contours of Constant Probability



Notes

$$\sigma_u > \sigma_v$$

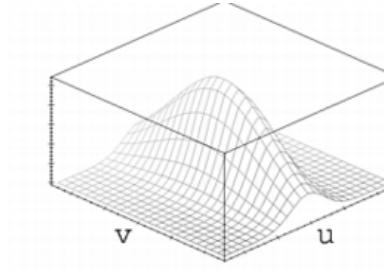
$$\rho_{uv} = 0$$

**Figure 4.12** Horizontal edge characteristics.

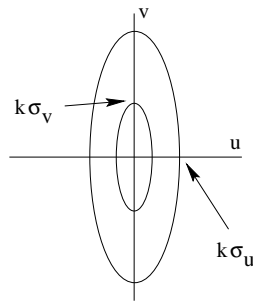
Covariance Matrix

$$\mathbf{R}_k = \begin{bmatrix} \sigma_u^2 & 0 \\ 0 & \sigma_v^2 \end{bmatrix}$$

Density Function



Contours of Constant Probability



Notes

$$\sigma_u < \sigma_v$$

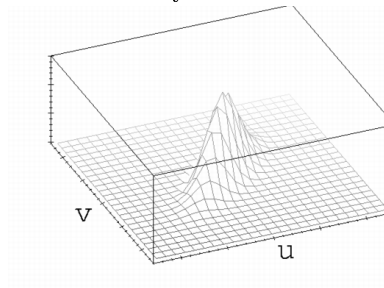
$$\rho_{uv} = 0$$

**Figure 4.13** Vertical edge characteristics.

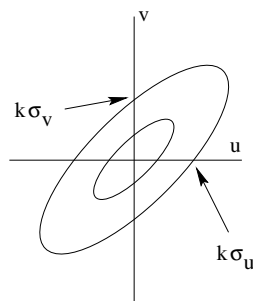
Covariance Matrix

$$\mathbf{R}_k = \begin{bmatrix} \sigma^2 & \rho_{uv}\sigma^2 \\ \rho_{uv}\sigma^2 & \sigma^2 \end{bmatrix}$$

Density Function



Contours of Constant Probability



Notes

$$\sigma = \sigma_u = \sigma_v$$

$$\rho_{uv} \neq 0$$

**Figure 4.14** Diagonal edge characteristics.

$\sigma_u \neq \sigma_v$ , as in Figures 4.12, 4.13, and 4.14, the location is more certain in one direction (given by the minor axis of the ellipse) than in the other direction (given by the major axis). As the length of the major axis approaches infinity, complete uncertainty on the location along this dimension is asserted. This uncertainty is the primary information needed to determine the spatial discrimination power of the SSDS for a particular case. Thus, it is sufficient to maintain the mean  $\mathbf{z}_k$  and covariance  $\mathbf{R}_k$  from a measurement. In the next section we explain how to estimate these quantities.

#### 4.2.6.2 Parameter estimation from the SSDS

This section describes a process for analyzing the SSDS to arrive at estimates for the mean and variance of a Gaussian random vector. The density function of this vector acts as an approximation to the response distribution (Equation (2.2)) for the purpose of tracking features.

The mode, or most probable value, of a random vector is located at the peak of the density function. We take the location of the minimum of the SSDS as our value for the mode of the vector,

$$\mathbf{z}_k = \operatorname{argmin}_{u,v} SSD(u, v). \quad (4.1)$$

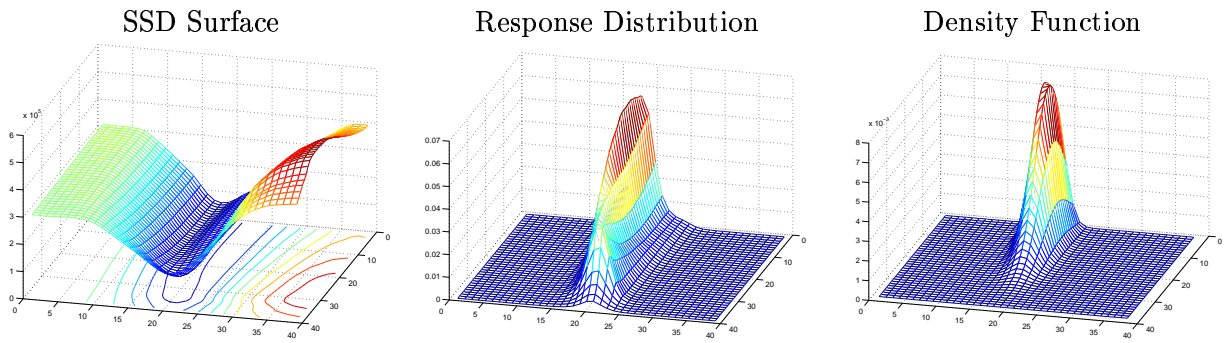
The variance of  $u$  ( $\sigma_u^2$ ), the variance of  $v$  ( $\sigma_v^2$ ), and the covariance between  $u$  and  $v$  ( $\rho_{uv}\sigma_u\sigma_v$ ) can be estimated directly from the response distribution using Equations (2.2) and (2.3), yielding the desired covariance matrix,

$$\mathbf{R}_k = \begin{bmatrix} \sigma_u^2 & \rho_{uv}\sigma_u\sigma_v \\ \rho_{uv}\sigma_u\sigma_v & \sigma_v^2 \end{bmatrix}, \quad (4.2)$$

which, as described above, contains complete information about the *orientation* and *shape* of the error ellipsoids. Figure 4.15 illustrates this process. Our computation of the normalization factor  $k$  in Equation (2.2) differs from [14]. We chose  $k$  such that

$$\sum_{u,v \in N} \mathcal{RD}(u, v) \approx 1. \quad (4.3)$$





**Figure 4.15** Approximation of response distribution by density function.

This has the effect of suppressing the off-peak response of the feature detector, and making the response distribution more closely approximate a Gaussian density function with the desired characteristics.

By computing the covariance as well as the variances, we retain information about the orientation of the ellipsoids of constant probability, as well as their intersection with the  $u$  and  $v$  axes. Therefore, we gain the ability to maintain information about directions of good spatial discrimination.

Of course, as we are only maintaining the mean and variance of the random vector, and not the complete SSDS, this is only an approximation to the complete information about local image structure given by the SSD. However, it does give some indication of both the absolute quality of the match and, in cases where edge features exist, the direction of the edge. Perhaps more importantly, we are explicitly recognizing the Gaussian assumptions made by the Kalman filtering framework (see Chapter 3).

### 4.3 Some Comments on Feature Selection

One interesting question to ask is, what is the minimum number of features needed to infer the joint angles of an articulated object? This is related to our problem of tracking the joint angles of an articulated object, but does not make the common assumption used in tracking

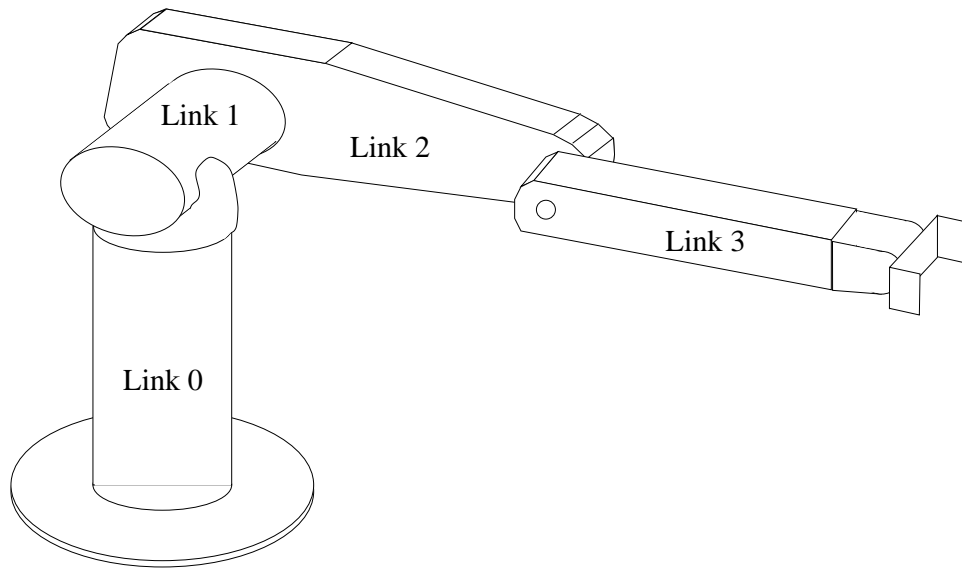
that the joint angles are in some local neighborhood of the previous joint angles. Without this assumption, the determination of the joint angles from point feature information reduces to a well-known research field known as *pose estimation*.

It has been shown that for the general case of estimating the position and orientation of a rigid object under perspective projection, unique solutions exist for 4 coplanar, but not collinear, points [100]. Thus, an upper limit on the minimum number of features required to compute the position and orientation of each link of an articulated object is four per active link of the object.

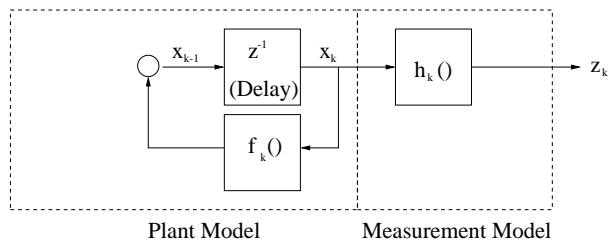
However, knowledge of the kinematics of the object could reduce this number. For example, consider the case of the PUMA robotic arm. If the first three joints of the arm are free to move, we refer to this configuration as having three degrees of freedom (3DOF). The estimation of the positions and orientation of each link independently therefore requires knowledge of the location in the image plane of 12 features.

We will show that consideration of the kinematic structure of the arm will reduce the minimum number of features to three. Haralick and Shapiro show [101] that the image plane locations of three collinear points under perspective projection are sufficient to recover the 3D parameters of the line containing those points. Therefore, if those points are located collinearly in known locations on link 3 of the PUMA (see Figure 4.16), this would be sufficient to recover the pose of link 3. The determination of the pose of link 3 also determines unique poses for links 1 and 2 of the arm. Joint angles  $q_0$ ,  $q_1$ , and  $q_2$  can be computed directly from these poses. With careful selection of features, three feature points can be used to infer the values of joint angles  $q_0$ ,  $q_1$ , and  $q_2$ .

As will be seen in Chapter 7, many more than four features are used for the case of a 3DOF PUMA robotic arm. Reasons for this include a more general formulation for the analysis of the kinematics (see Chapter 3), compensation for the possibility of occlusion, or missing features (see Section 4.5), the possibility of feature measurements that do not yield accurate feature locations in all directions (see Chapter 5), and the possibility of a singularity in the observation



**Figure 4.16** PUMA robotic arm.



**Figure 4.17** Reduced system model.

function (the line containing the three points could intersect the optic center of the camera, yielding identical feature locations for the points).

Chapter 3 discussed estimation of the state of a nonlinear system from noisy observations of that system. As Section 4.1 explains, we use this system model in our object tracking system. System theory defines *observability* as the ability to estimate the state of a system from the knowledge of the input to the system, models of the system dynamics, and the output of the system. Note that the noise assumed in our system model Equation (3.1) is not tolerated in basic system theory, and will be neglected for the remainder of this section.

Under the assumption of no control input and no additive noise, the system is of the form given in Figure 4.17. Then it can be seen (see [102]) that the determination of the *observability* of the system reduces to a test of the linear independence of the columns of the image Jacobian,  $\frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k}$ . This observation allows some comments on the location of features in our system.

First, each joint angle to be estimated must affect at least one element of the measurement vector. That is, at least one feature point must move with each joint angle of interest. Note that this does not imply that a feature point must be present on each link of the arm, although this is a common case. As shown above, determination of the position and orientation of the arm are sufficient to infer all three joint angles in the 3DOF case.

Second, the effects of singularities in the image Jacobian can be determined. The observation function  $\mathbf{h}_k(\mathbf{x}_k)$  (see Section 7.4) models the mapping from the joint angles of the robot to feature points in the image. The matrix derivative of this function,  $\frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k}$ , is referred to as the image Jacobian [103]. The observation function is the composition of the forward kinematics of the robot, the point locations on the robot that project to the feature points, and the projection equations. The image Jacobian is therefore the composition of the arm Jacobian and the projection Jacobian. Due to this, singularities in *either* the arm Jacobian *or* the projection Jacobian cause singularities in the image Jacobian.

In physical terms, singularities in the arm Jacobian occur in configurations where incremental motion in a joint angle leads to *no incremental motion* in the workspace location of a point on the robot. Singularities in the projection Jacobian occur when incremental motion of a point in the workspace leads to *no incremental motion* in the image feature. In either of these cases, movement of the relevant feature reveals no useful information about motion of the relevant joint.

If a singularity occurs in the image Jacobian, and if there is no other feature that reveals information about motion about the relevant joint angle, that column in the image Jacobian becomes zero. In system theory, this is referred to as a mode of the state vector becoming *unobservable*. Note that if there is another feature whose incremental motion reveals information

about the relevant joint angle, the Jacobian does not drop rank, and the mode of the system remains observable.

Nulls in the image Jacobian can occur for reasons besides the singularities described above. Features which are self-occluded (see Section 4.5.2) are represented as a constant in the observation function, and thus cause nulls in the Jacobian. This is another justification for our use of several redundant features for each joint angle on the arm.

## 4.4 Assimilation of Feature Tracking Results

In this section, we describe the combination of tracking results from individual features into a single system feature tracking result. The use of feature tracking results to update object models was introduced in Section 2.6. In this section, we describe the case where feature tracking results are available for every feature. In Section 4.5 we describe the modifications required to account for missing feature measurements. We then describe the use of the system feature tracking result in the overall object tracking system.

Once  $\mathbf{z}_k$  and  $\mathbf{R}_k$  have been computed for each feature (in this section, we will denote individual feature measurements by  $\mathbf{z}_k^f$  and their covariance by  $\mathbf{R}_k^f$ ), the information from the individual feature trackers must be combined to make observation and covariance vectors for the system as a whole (denoted as  $\mathbf{z}_k$  and  $\mathbf{R}_k$ ).

To make this combination, we concatenate the means of the individual vectors,

$$\mathbf{z}_k = \begin{bmatrix} \mathbf{z}_k^1 \\ \vdots \\ \mathbf{z}_k^F \end{bmatrix},$$

where there are  $F$  features. If we make the assumption that the additive noise in the feature measurements is independent, the covariance of  $\mathbf{z}_k$  can be computed by constructing a block-

diagonal matrix from the individual covariance matrices,

$$\mathbf{R}_k = \begin{bmatrix} \mathbf{R}_k^1(1,1) & \mathbf{R}_k^1(1,2) & \cdots & 0 & 0 \\ \mathbf{R}_k^1(2,1) & \mathbf{R}_k^1(2,2) & & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & & \mathbf{R}_k^F(1,1) & \mathbf{R}_k^F(1,2) \\ 0 & 0 & \cdots & \mathbf{R}_k^F(2,1) & \mathbf{R}_k^F(2,2) \end{bmatrix}$$

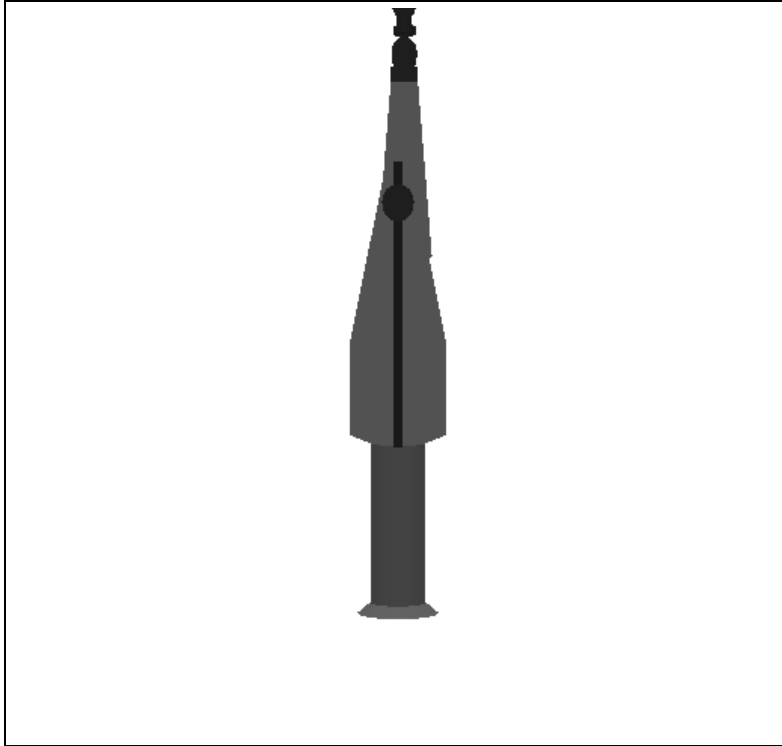
where the individual  $\mathbf{R}_k^f$  are as described above. Modeling the interactions between the uncertainties of the individual feature trackers is a nontrivial task [14] and is a topic for future research. The interactions of the locations of the features are modeled, in that the movement of each feature plays a role in updating the state estimates  $\hat{\mathbf{x}}_k$ , as described in Chapter 3.

## 4.5 Treatment of Feature Occlusion

In this section, we describe how the system treats feature occlusion. Feature occlusion is defined as any situation where a feature is not completely visible from the viewpoint of the camera. Whether a particular feature is occluded is a function of the camera viewpoint, the object configuration, and the position of external objects that may obstruct the view of the object. We differentiate between three types of occlusion and describe the treatment of each type. Note that in an object-tracking framework, the loss of a feature due to occlusion causes mistracking only to the extent that the erroneous feature tracking information causes errors in the state estimate. Since an overconstrained system can compute state estimates with some missing observations, if a feature which is missing due to occlusion becomes visible it is automatically reacquired. This is one of the benefits of using object models in object tracking.

### 4.5.1 Features off-screen

The object geometric model and imaging model can be used to compute the 3D location of each feature and the location of the projection of that point onto the image plane. If that projection is outside the finite portion of the image plane actually observed, that feature is

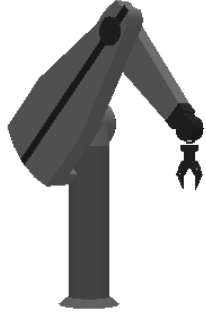


**Figure 4.18** Example of off-screen features.

said to be *off-screen*. Features determined to be off-screen are marked not visible. Figure 4.18 illustrates a configuration in which the gripper fingers would be determined to be off-screen.

#### 4.5.2 Self-occlusion of features

Self-occlusion is occlusion of a feature by another part of the modeled object. To detect self-occlusion, the 3D location of each feature point and the location of the projection of that point onto the image plane is computed as described in Section 4.5.1. During the rendering process, the depth of the closest object to the image plane at each pixel is recorded in a special buffer termed the *z-buffer*. This is a well known method for 3D rendering in computer graphics [104]. To compute self-occlusion, the depth recorded in the *z-buffer* at each feature projection point is compared against the computed distance from the image plane for that feature to determine if the feature is visible at the given configuration. A feature is visible in a given configuration



Gripper not occluded



Gripper self-occluded

**Figure 4.19** Example of self-occlusion.

if its depth is equal to that recorded in the appropriate portion of the  $z$ -buffer. An example of a configuration where self-occlusion occurs is shown in Figure 4.19.

If a feature  $f$  is found to be not visible due to being off-screen or due to self-occlusion, the relevant elements of the *observation function* ( $h_k^{2f}(\mathbf{x}_k)$  and  $h_k^{2f+1}(\mathbf{x}_k)$ ) are set to a constant. As seen in Section 3.2, this will cause

$$\frac{\partial h_k^{2f}}{\partial \mathbf{x}_k}(\hat{\mathbf{x}}_k) = \frac{\partial h_k^{2f+1}}{\partial \mathbf{x}_k}(\hat{\mathbf{x}}_k) = 0,$$

and results from tracking the nonvisible feature will not be used in the state update. Indeed, in the interest of performance, tracking is not even performed on nonvisible features.

### 4.5.3 External occlusion of features

External occlusion is occlusion that cannot be predicted from the object model and estimated configuration. This includes features predicted to be visible that are not visible, as well as features that have some unmodeled object obscuring the visibility of the feature. For example, if someone walks between the camera and a feature of interest (see Figure 4.20), the





Gripper not occluded



Gripper externally occluded

**Figure 4.20** Example of external occlusion.

resulting measurement would be termed *externally occluded*. The system does not have a test for external occlusion, such as the tests described above.

When a feature is externally occluded, the template will not match any area of the image very well. Thus, there is not a single well defined peak as shown in Figures 4.7 and 4.8. When the variance and covariance numbers are estimated as described in Chapter 5,  $|\mathbf{R}_k|$  will be large, indicating high spatial uncertainty on the returned feature location.

Since the covariance of a measurement is used when updating the state estimates, measurements with high uncertainty will be selectively ignored (when compared to measurements with low uncertainty). Therefore, the assumed dynamic model will be used to a greater extent when updating the state estimate.

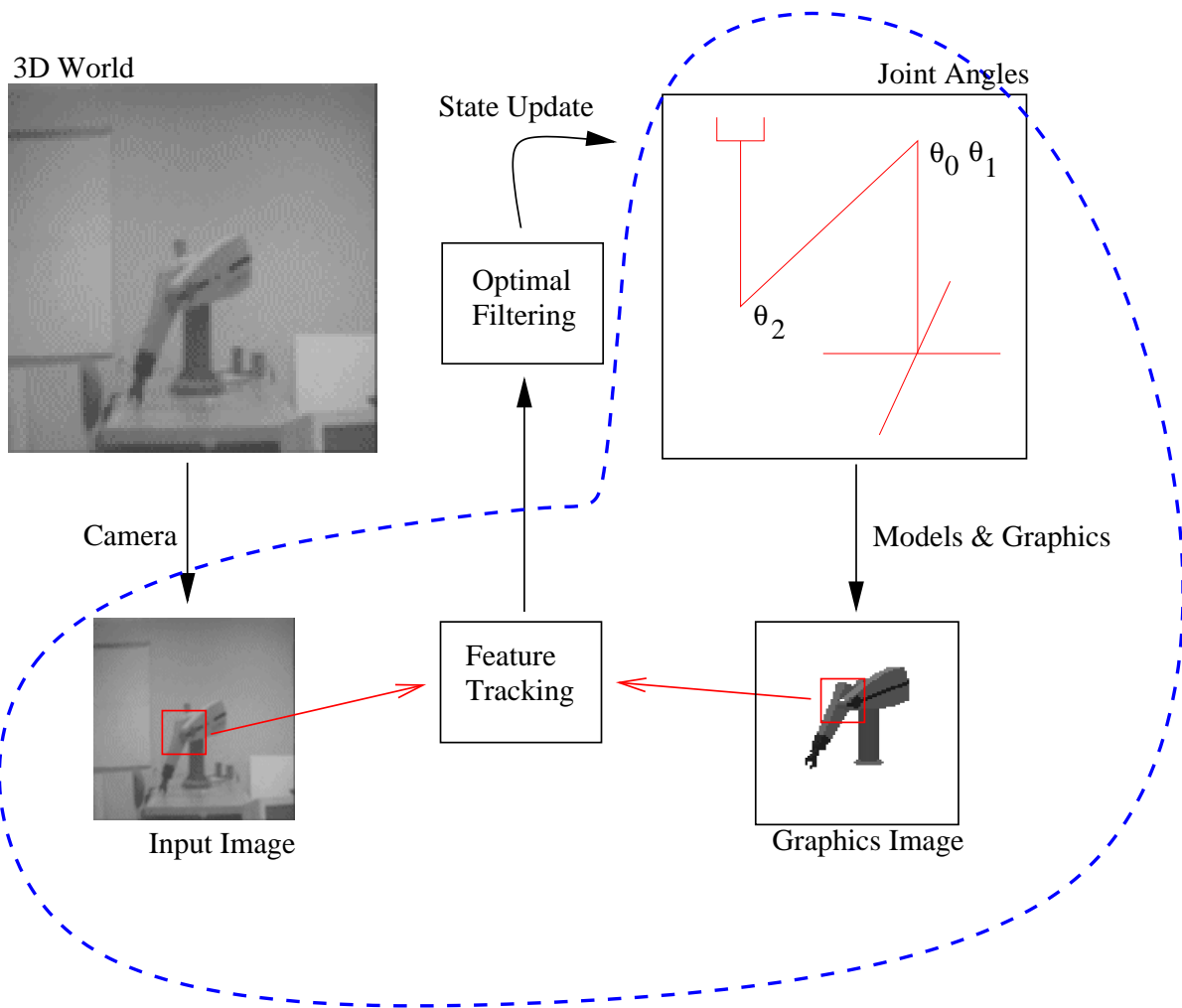
## CHAPTER 5

### APPLICATION OF FEATURE TRACKING

In this chapter, we will illustrate the use of feature appearance templates in locating the most likely position of a feature in an image. Figure 5.1 shows the portion of the tracking system discussed in this chapter. We will illustrate the use of the sum-of-squared-differences-surface (SSDS) described in Chapter 4 to compute an estimate of the feature location and the spatial uncertainty associated with that location.

In Section 4.2.3 we described the creation of *templates* for individual features. A template describes the expected appearance of a feature, and we use a different template for each visible feature. We currently use fixed size rectangular templates ( $39 \times 39$  pixels in the current implementation) and do not attempt to separate the effect of the background from the effect of the feature of interest. Ideally, some knowledge about the physical size of the feature would be combined with the imaging model to estimate the size and shape of the feature of interest in the image, and this knowledge would be used to determine the size and shape of the feature template.

Using the models described in Section 4.2.1, an optimal estimate for the location in the image plane of each feature is computed. We define a fixed size rectangular search region ( $39 \times 39$  pixels in the current implementation) about this location, and compare the area around each pixel in the search region to the area around the center pixel of the template, using the SSD dissimilarity measure as described in Section 4.2.4. A topic for future research would be to use the models described in Section 4.2.1 and the uncertainties  $\mathbf{P}_k$  associated with the state



**Figure 5.1** Portion of tracking system discussed in this chapter.

vector  $\mathbf{x}_k$  (review Chapter 3 for definitions of these terms) to compute the area in the image containing the majority of the projected probability mass associated with the measurement. Kosaka and Kak [39] implement an idea similar to this for line tracking in their system. The result is that the computational effort spent computing SSD measures is concentrated in the region where the feature is most likely to appear in the image.

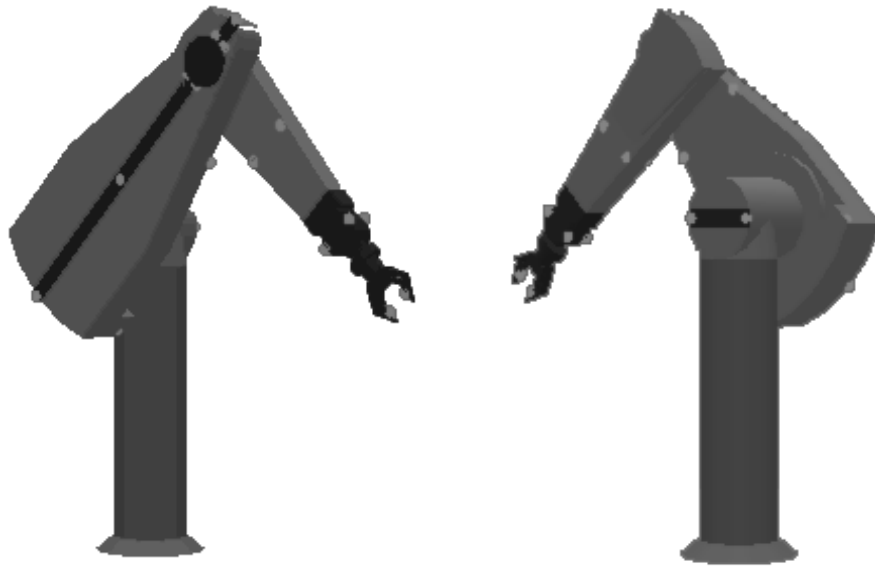
In the following sections, we will illustrate the use of the framework described in Chapter 4 for feature tracking. In this chapter, we do not describe the use of the feature tracking results generated, but concentrate on the feature tracking. Chapter 6 describes the use of feature tracking results to update state estimates. A context for the feature locations described in the following sections can be seen in Figure 5.2, which depicts the locations of features on the arm. We begin in Section 5.1 by showing the feature tracking for a gripper finger, typically providing enough spatial discrimination to approximate a point feature. In Section 5.2 we show feature tracking results for a point on an edge of the arm. Since there is an edge at this point, only one dimension of spatial discrimination is recovered from this feature tracking. Finally, in Section 5.3 we illustrate a point feature that acts, in certain configurations of the robot, like an edge feature.

## 5.1 Point Feature: Gripper Finger

The feature appearance template and search region for this case are shown in Figure 5.3. Note that since the SSD measure involves the image area surrounding a pixel, a border around the search region must be retained for each search region.

The results of the feature tracking are shown in Figure 5.3(d)-(f). The cross indicates the location of the minimum point of the SSD surface. The complete SSDS and the Gaussian approximation to this surface are shown in Figure 5.3(e)-(f).

A 2D measurement and  $2 \times 2$  covariance measurement are the output of the feature tracking, and are used directly in the EKF framework described in Chapter 3.



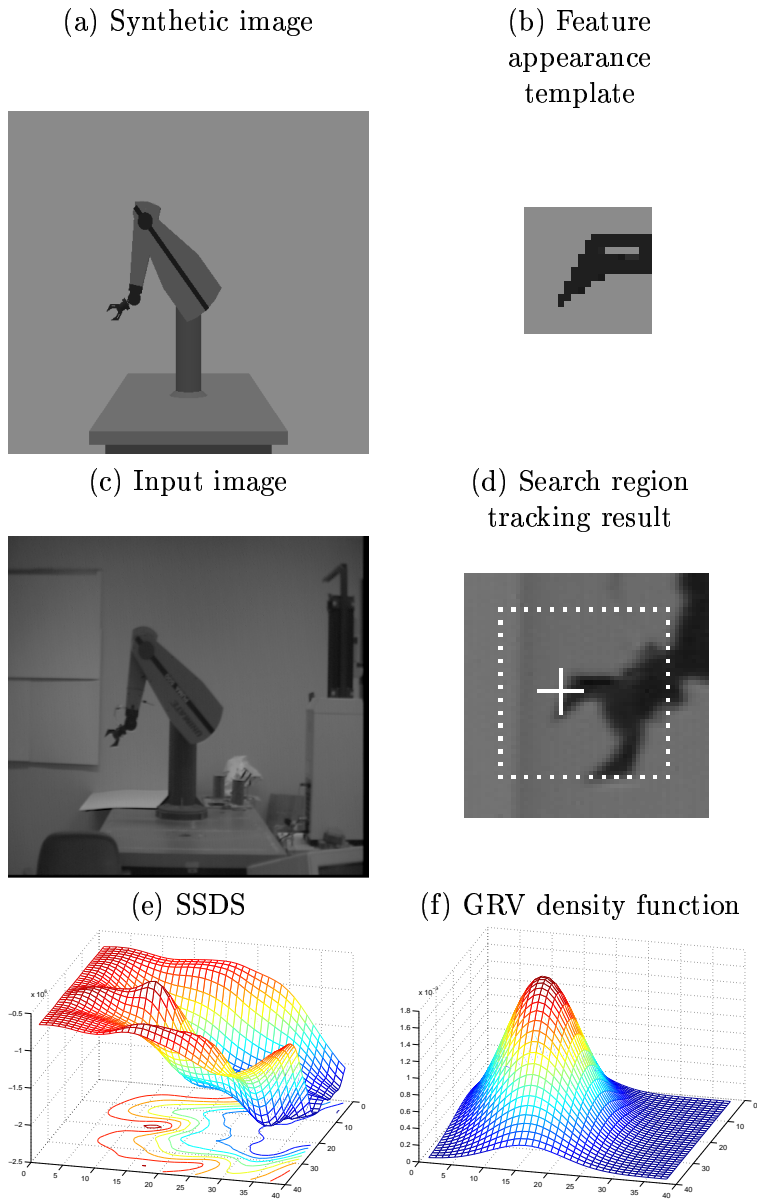
**Figure 5.2** Features tracked in this section.

## 5.2 Edge Feature

The feature appearance template and search region for this case are shown in Figures 5.4-5.7. Note that the orientation of the edge in the feature depends on the estimated configuration of the robot  $\hat{\mathbf{x}}_k$ .

As the estimated configuration of the robot  $\hat{\mathbf{x}}_k$  changes, the direction of the edge projected onto the image plane will change. Thus, the direction in which this template discriminates will change. Note that the system retains full information about the direction of maximum spatial discrimination for use during state estimate update.

The results of the feature tracking for the edge feature are shown in Figures 5.4-5.7. The cross in (d) for each figure indicates the minimum point of the SSD surface. The complete SSDS and the Gaussian approximation to this surface are shown in (e)-(f) for each figure.



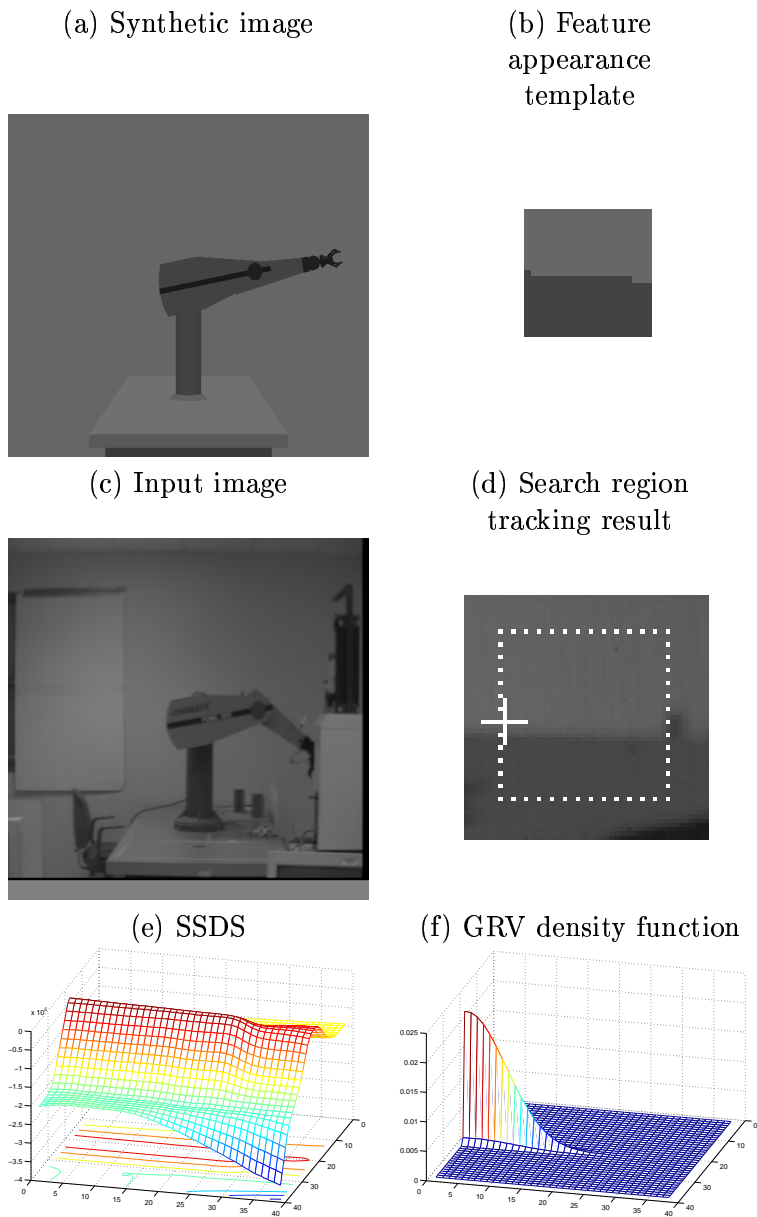
Notes

$$\sigma_u = 7.4774, \quad \sigma_v = 12.4664$$

$$\mathbf{z}_k = [23 \quad 28]^T$$

$$\mathbf{R} = \begin{bmatrix} 55.9111 & 20.8417 \\ 20.8417 & 155.4108 \end{bmatrix}$$

**Figure 5.3** Tracking results for gripper feature.



Notes

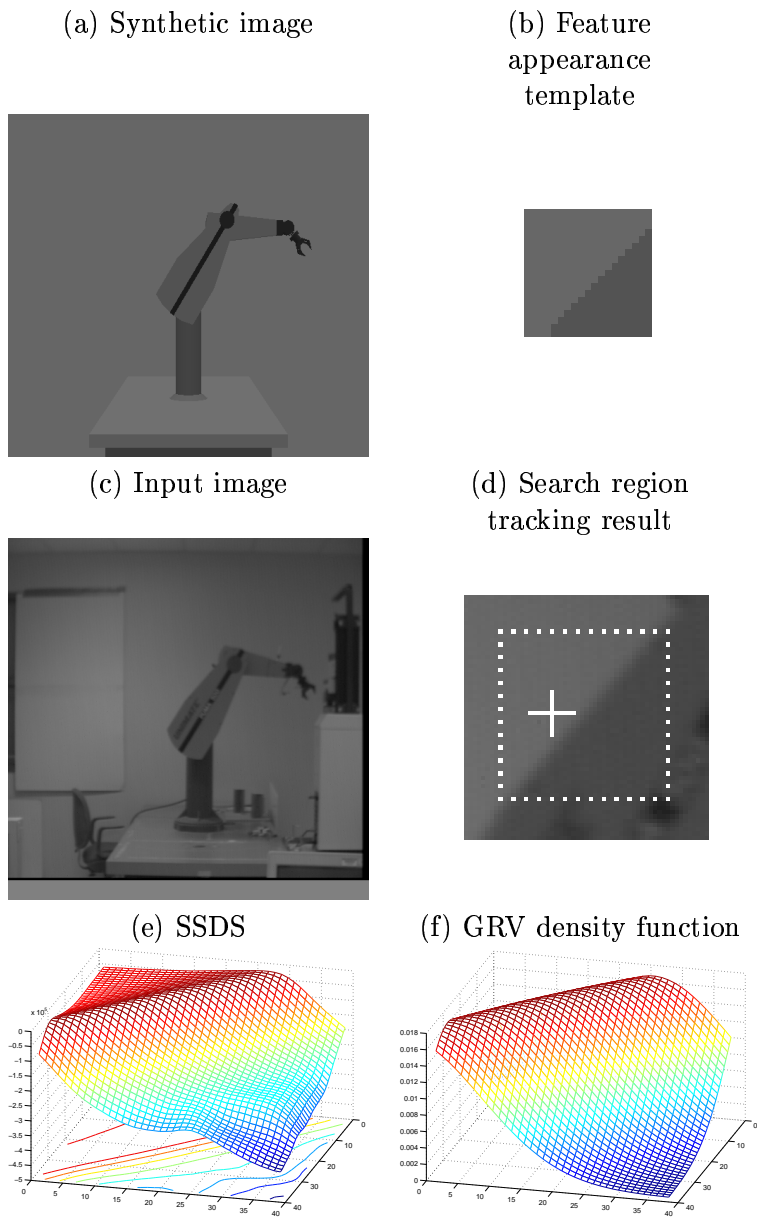
$$\hat{\mathbf{x}}_k^2 = -170$$

$$\sigma_u = 7.3565, \quad \sigma_v = 0.6812$$

$$\mathbf{z}_k = [10 \quad 30]^T$$

$$\mathbf{R} = \begin{bmatrix} 54.1183 & 0.7824 \\ 0.7824 & 0.4641 \end{bmatrix}$$

**Figure 5.4** Tracking results for edge feature.



Notes

$$\hat{\mathbf{x}}_k^2 = -135$$

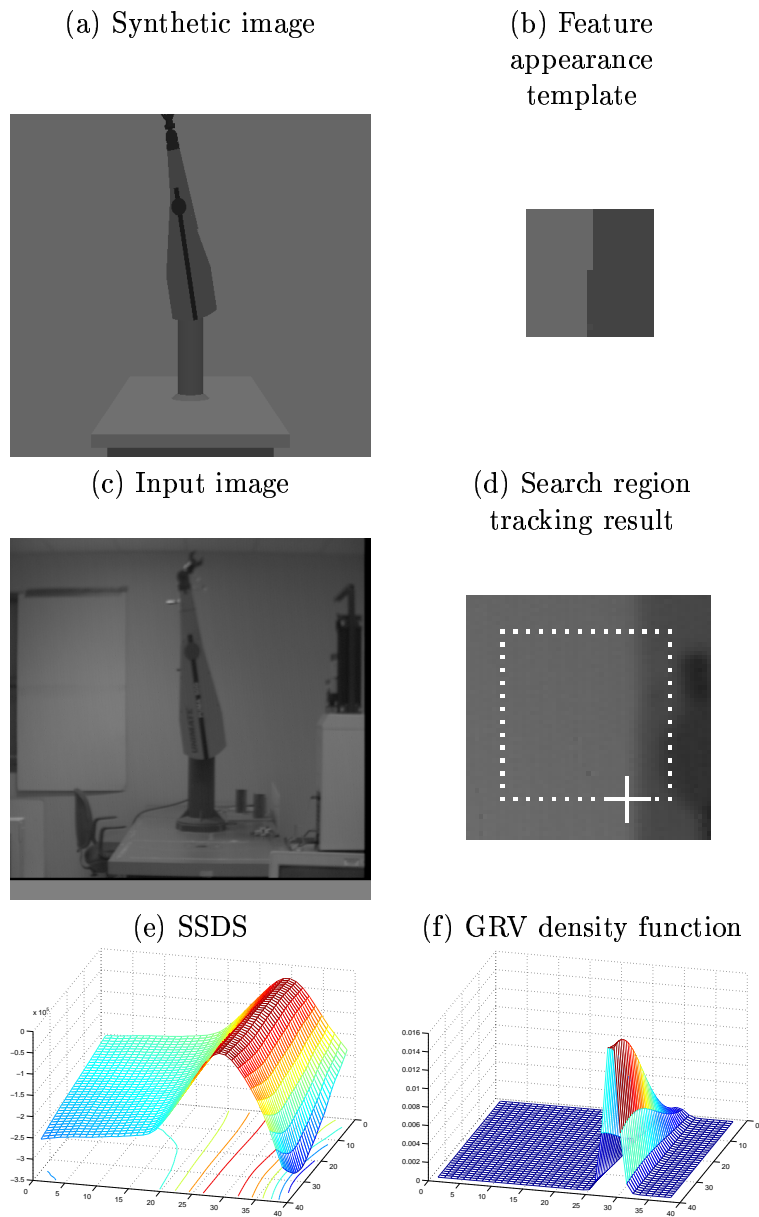
$$\sigma_u = 6.2511, \quad \sigma_v = 7.6098$$

$$\mathbf{z}_k = [21 \quad 28]^T$$

$$\mathbf{R} = \begin{bmatrix} 39.0761 & -46.6814 \\ -46.6814 & 57.9095 \end{bmatrix}$$

**Figure 5.5** Tracking results for edge feature.





Notes

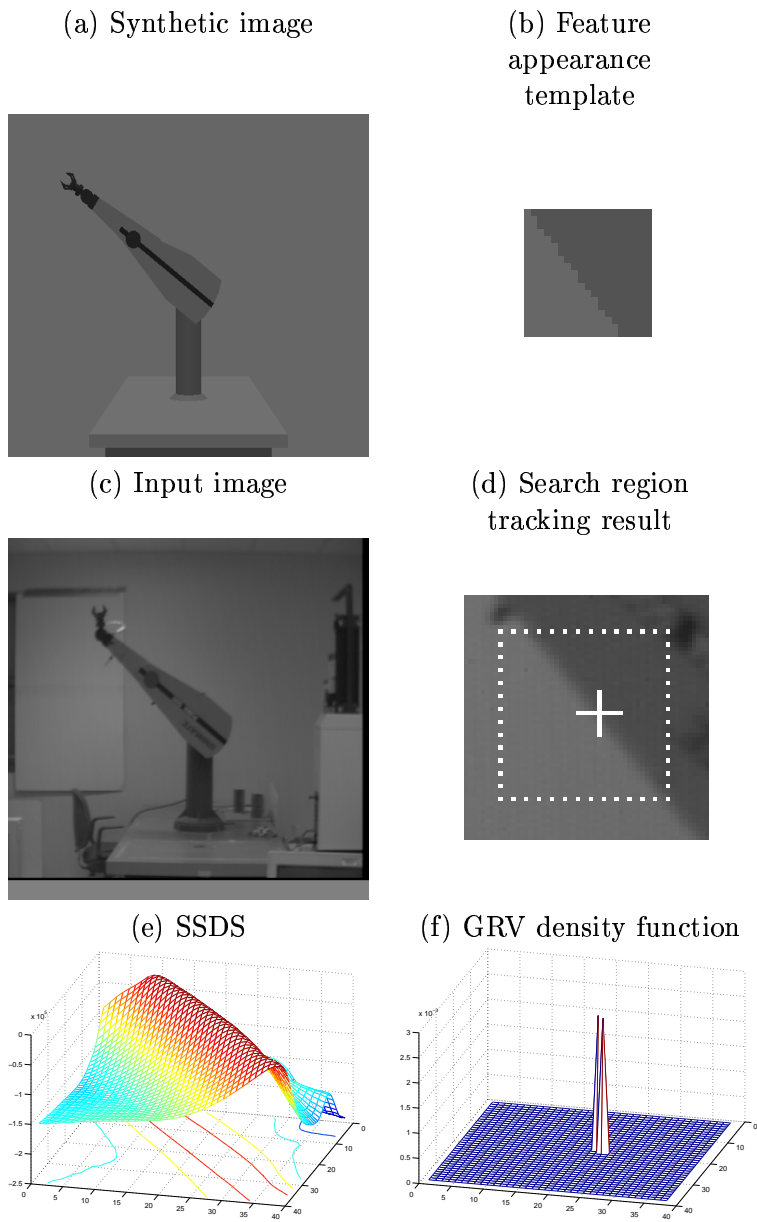
$$\hat{\mathbf{x}}_k^2 = -80$$

$$\sigma_u = 0.8159, \quad \sigma_v = 11.6058$$

$$\mathbf{z}_k = [38 \quad 48]^T$$

$$\mathbf{R} = \begin{bmatrix} 0.6657 & -2.8128 \\ -2.8128 & 134.6939 \end{bmatrix}$$

**Figure 5.6** Tracking results for edge feature.



Notes

$$\hat{\mathbf{x}}_k^2 = -35$$

$$\sigma_u = 7.1511, \quad \sigma_v = 9.0809$$

$$\mathbf{z}_k = [32 \quad 28]^T$$

$$\mathbf{R} = \begin{bmatrix} 51.1386 & 64.3817 \\ 64.3817 & 82.4623 \end{bmatrix}$$

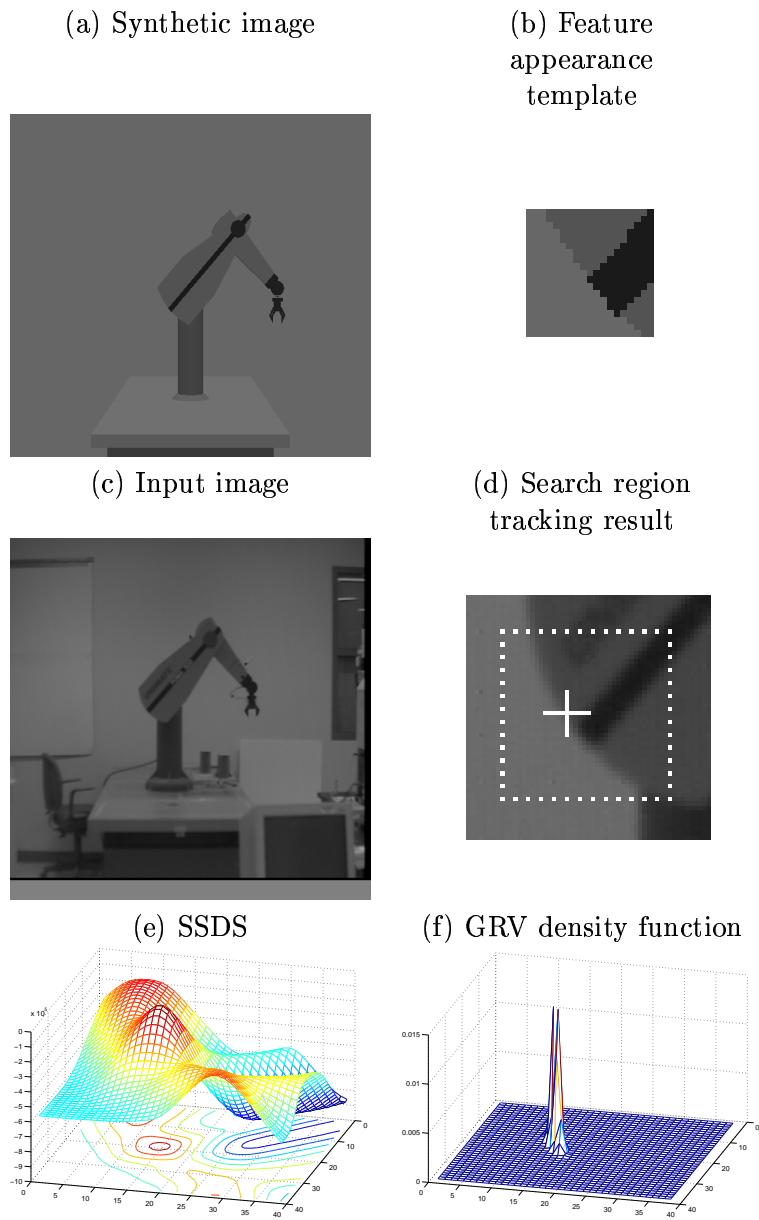
**Figure 5.7** Tracking results for edge feature.

### 5.3 Degenerate Point Feature

In this section, we illustrate the usefulness of on-line estimation of template efficacy. The feature described in this section is a point of high texture in both directions when the surface that the point is on is approximately parallel to the image plane. When the surface on which the point is painted is parallel to the image plane, the feature acts like a point feature. However, since our framework tracks under a wide variety of object configurations, the appearance of features can change dramatically during tracking. For example, this feature can act like an edge feature in certain configurations. Figures 5.8 and 5.9 illustrate this feature of the tracking system. Since  $\mathbf{R}$  is used in assimilating feature tracking results, information about the quality of the feature measurement will be used along with the measurement.

### 5.4 Externally Occluded Point Feature

In this section, we present another illustration of the usefulness of on-line estimation of template efficacy. This feature has the same template as in Section 5.1, except that the feature has been occluded by the person in Figure 5.10(a). As described in Section 4.5.3, this mismatch causes larger values for the variances, and the measurement returned is devalued accordingly. Since  $\mathbf{R}$  is used in assimilating feature tracking results, this information about the feature will be weighted lightly in the state update.



Notes

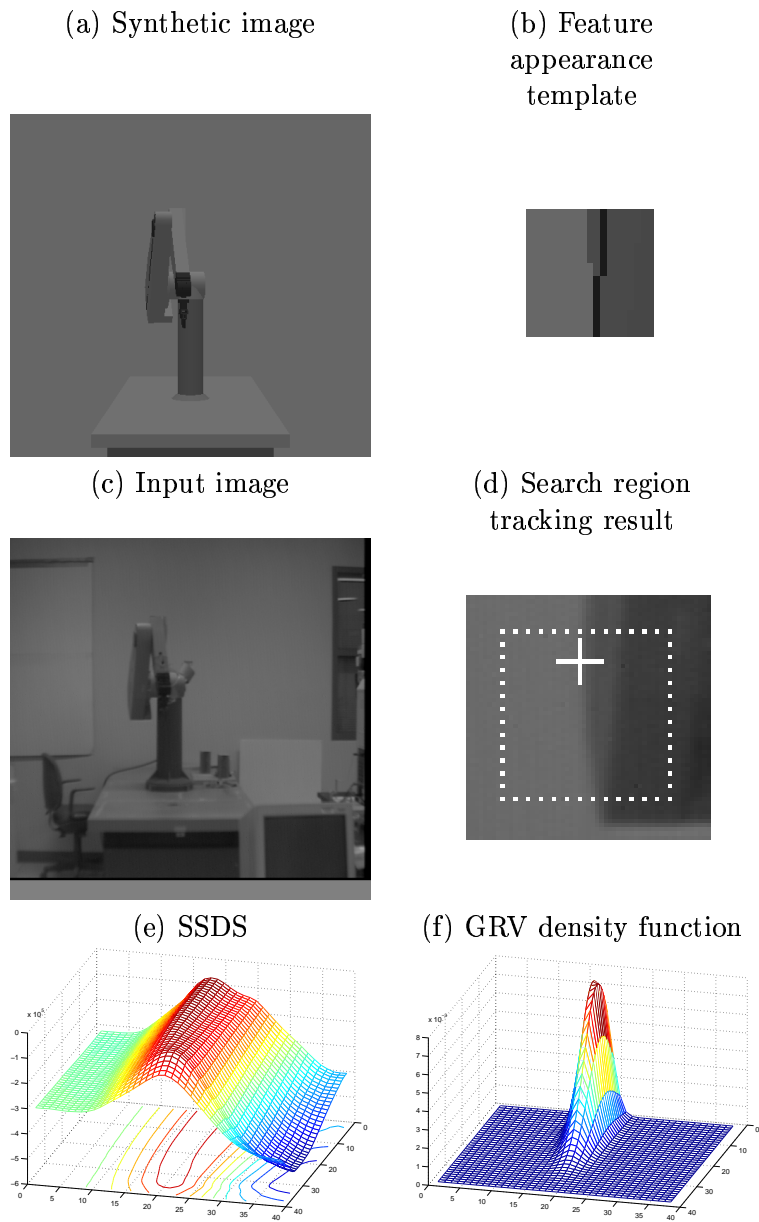
$$\hat{\mathbf{x}}_k^0 = 0$$

$$\sigma_u = 3.0798, \quad \sigma_v = 5.6748$$

$$\mathbf{z}_k = [24 \quad 28]^T$$

$$\mathbf{R} = \begin{bmatrix} 9.4854 & 15.2166 \\ 15.2166 & 32.2039 \end{bmatrix}$$

**Figure 5.8** Tracking results for nondegenerate point feature.



Notes

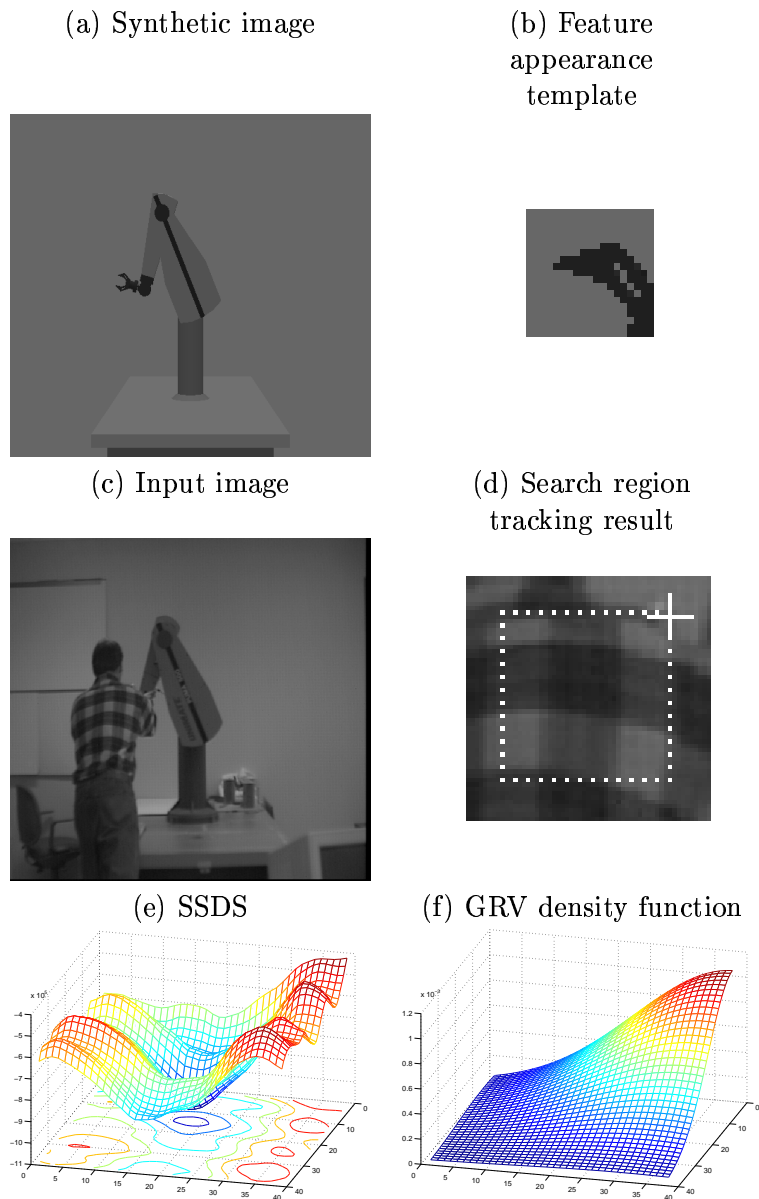
$$\hat{\mathbf{x}}_k^0 = -85$$

$$\sigma_u = 1.5383, \quad \sigma_v = 9.6682$$

$$\mathbf{z}_k = [27 \quad 16]^T$$

$$\mathbf{R} = \begin{bmatrix} 2.3663 & 5.0629 \\ 5.0629 & 93.6683 \end{bmatrix}$$

**Figure 5.9** Tracking results for degenerate point feature.



Notes

$$\hat{\mathbf{x}}_k^0 = \mathbf{0}$$

$$\sigma_u = 13.3954, \quad \sigma_v = 10.2426$$

$$\mathbf{z}_k = [47 \quad 0]^T$$

$$\mathbf{R} = \begin{bmatrix} 179.4363 & -9.1616 \\ -9.1616 & 104.9117 \end{bmatrix}$$

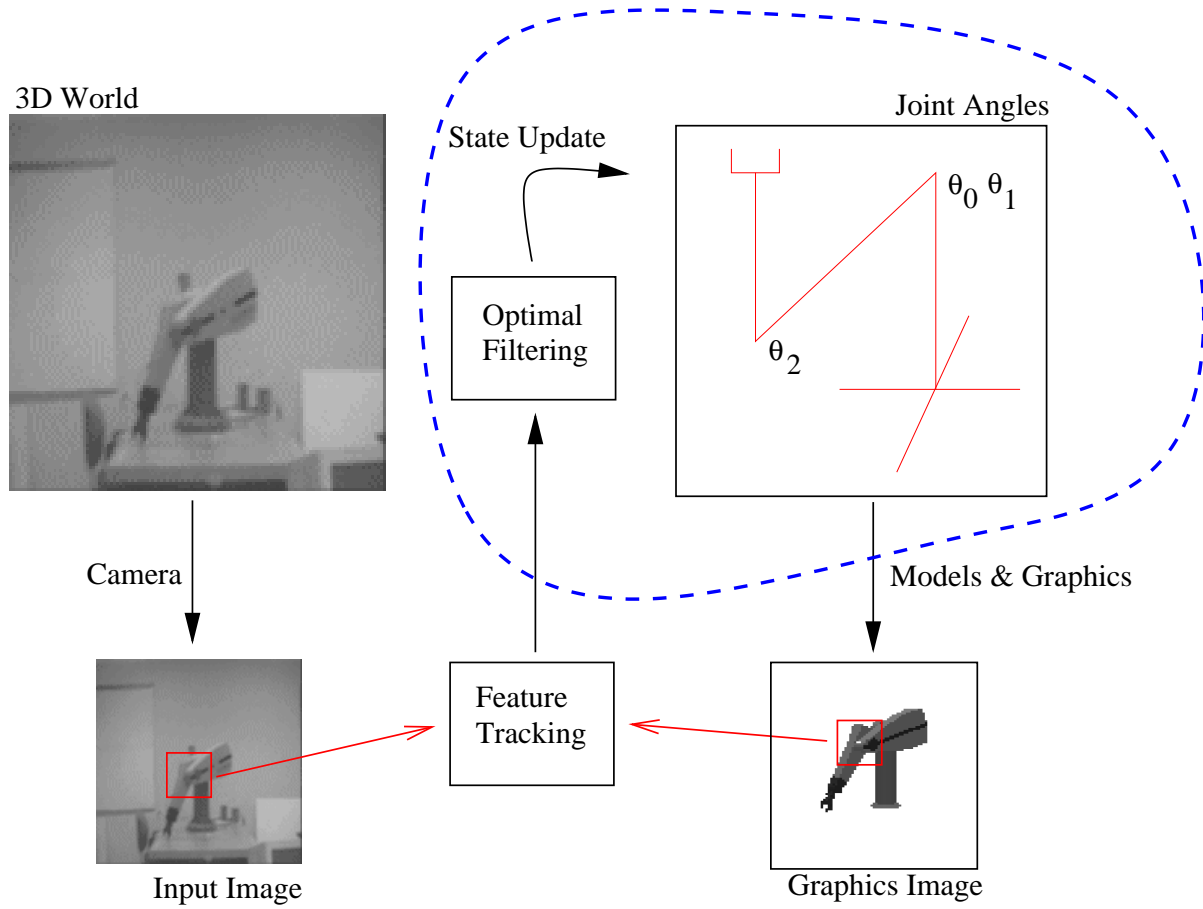
**Figure 5.10** Tracking results for externally occluded point feature.

## CHAPTER 6

### APPLICATION OF KALMAN FILTERING FRAMEWORK

In this chapter, we will apply feature tracking results such as those described in Chapter 5 within the tracking system presented in Chapter 4 to several different object tracking scenarios. We will begin with some simple examples, thus illustrating the basic behavior and properties of the framework. Figure 6.1 shows the portion of the tracking system discussed in this chapter. Some more complicated scenarios will then be examined, illustrating the power and extendibility of the system.

We begin by presenting and analyzing the system for a one-link arm being observed by a 1D orthogonal sensor. In this case, we assume the simplest dynamic model, that of constant position motion. Arm movement is tolerated by modeling it as noise  $\mathbf{w}_k$  injected into the state vector  $\mathbf{x}_k$  at each time step. We then show the changes in the filter when using an elevated 2D perspective sensor for the same arm. Next, we present a two-link arm being observed by a 2D perspective sensor, again with a constant position dynamic model, in order to illustrate the effect of more complicated geometric models on the tracking system. Fourth, we present the effect of a constant joint space velocity dynamic model on the system, by analyzing the filter for a one-link arm under this model. Fifth, we examine the major case of interest: an arm with three degrees of freedom being observed by a 2D perspective sensor, under a constant joint space velocity dynamic model. Finally, we present an extension to the basic framework, where the geometric model is incompletely specified. In this case, we track a two-link arm being observed by a 2D orthogonal sensor, where the link lengths are only approximately known.



**Figure 6.1** Portion of tracking system discussed in this chapter.

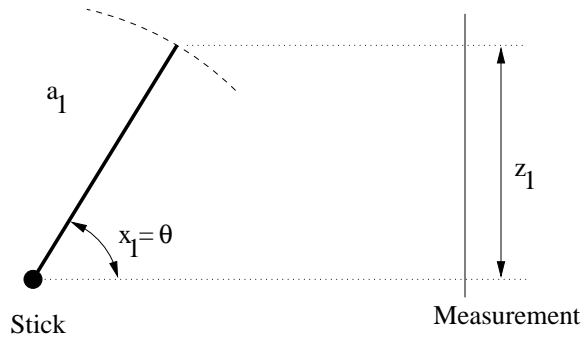


**Table 6.1** Models used in Case 1.

Object geometric model	1DOF one-link arm
Dynamic model	Constant position
Imaging model	1D orthogonal projection

## 6.1 Case 1: The Base Case

In this example, illustrated in Figure 6.2, there is a planar one-link arm that can rotate about one end that is fixed at the origin. The models used in this case are shown in Table 6.1. The configuration space in this example is 1D. We assume that the robot is embedded in a 2D workspace, and there is an orthographic projection onto the 1D sensor plane (to the right of the robot, sufficiently far away that there is no interaction between the arm and the sensor plane, at least a distance  $a_1$ ).



**Figure 6.2** Case 1.

### 6.1.1 System and filter definition

Using the notation outlined in Section 3.2, the (scalar) filter for this problem is as follows. The derivation of the equations presented in this section can be found in Appendix A. This system has one parameter, so the state vector has one entry,

$$x_k = \theta. \tag{6.1}$$

With a constant position dynamic model, the prediction equations are

$$f_k(x_k) = x_k \quad (6.2)$$

$$\frac{\partial f_k}{\partial x_k} = 1. \quad (6.3)$$

Using orthogonal projection, the observation equations are

$$h_k(x_k) = a_1 \sin(x_k) \quad (6.4)$$

$$\frac{\partial h_k}{\partial x_k} = a_1 \cos(x_k)$$

We assume for simplicity that the process noise does not vary with the state,

$$G'_k(x_k) \equiv 1.$$

Under these assumptions, the filter equations are as follows. Recall that  $\mathbf{Q}_k$  is the covariance matrix for the system noise and that  $\mathbf{R}_k$  is the covariance matrix for the observation noise. In this case, both  $\mathbf{Q}_k$  and  $\mathbf{R}_k$  are scalar.

$$P_{k,k-1} = P_{k-1,k-1} + Q_k \quad (6.5)$$

$$\hat{x}_{k|k-1} = \hat{x}_{k-1|k-1} \quad (6.6)$$

$$K_k = P_{k,k-1} \frac{a_1 \cos(\hat{x}_{k-1})}{a_1^2 \cos^2(\hat{x}_{k-1}) P_{k,k-1} + R_k} \quad (6.7)$$

$$P_{k,k} = P_{k,k-1} (1 - K_k a_1 \cos(\hat{x}_{k-1})) \quad (6.8)$$

## 6.1.2 Filter analysis

To capture the usefulness of this model for tracking, it is helpful to observe the response of the filter components to changes in the configuration of the robot. We will investigate the relative weighting of estimates and observations, then observe the changes in filter components as the filter converges to an estimate over time.

### 6.1.2.1 Weighting of predictions and observations

Recall the intuitive interpretation of the Kalman filtering update equations given in Section 3.4. In this interpretation, two measurements  $z_1$  and  $z_2$  of a random variable  $x$  were

combined to arrive at  $\hat{x}$ , an optimal estimate for  $x$ . Weighting factors based on the relative uncertainty of each measurement were used in the combination. These weighting factors vary between zero and unity, and determine the relative importance of each measurement in the overall estimate. We also presented the analogy between this combination of two measurements and the measurement update portion of the Kalman filter. In the Kalman filter, a combination of the observation prediction and the actual observation becomes the new observation estimate.

In this example  $h_k(\hat{x}_k)$  is analogous to  $z_1$ , with uncertainty  $\frac{\partial h_k}{\partial x_k} P_k \frac{\partial h_k}{\partial x_k}^T$ . Then  $z_k$  is analogous to  $z_2$ , with uncertainty  $R_k$ . The weighting factor for  $z_1$  becomes

$$\frac{\sigma_{z_2}^2}{\sigma_{z_1}^2 + \sigma_{z_2}^2} = \frac{a_1^2 \cos^2(\hat{x}_{k-1}) P_{k,k-1}}{a_1^2 \cos^2(\hat{x}_{k-1}) P_{k,k-1} + R_k},$$

which will be referred to as the *observation weighting factor*, or OWF. The OWF determines the relative importance of the observation (relative to the prediction) in the filter update and varies between zero and one in the scalar case. More generally, the OWF can be determined directly from the Kalman filtering equations as

$$OWF = \left[ \frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_{k-1}}(\hat{\mathbf{x}}_{k|k-1}) \right] \mathbf{K}_k.$$

The weighting factor for  $z_2$  then becomes

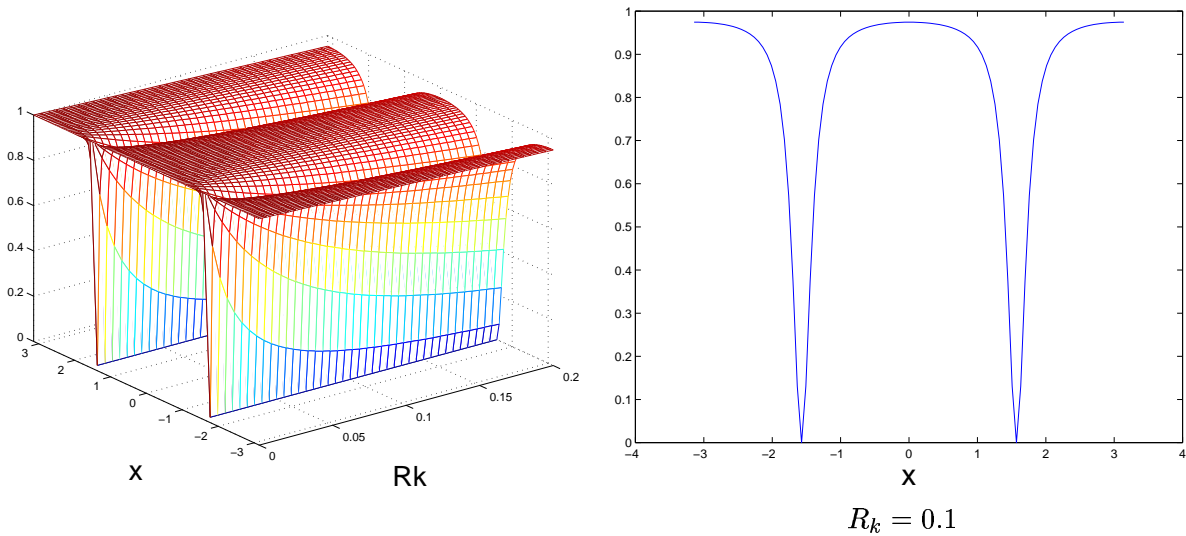
$$\frac{\sigma_{z_1}^2}{\sigma_{z_1}^2 + \sigma_{z_2}^2} = \frac{R_k}{a_1^2 \cos^2(\hat{x}_{k-1}) P_{k,k-1} + R_k},$$

which will be referred to as the *prediction weighting factor*, or PWF. More generally, the PWF can be determined directly from the Kalman filtering equations as

$$PWF = \mathbf{R}_k \left[ \frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_{k-1}}(\hat{\mathbf{x}}_{k|k-1}) \mathbf{P}_{k,k-1} \frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_{k-1}}(\hat{\mathbf{x}}_{k|k-1})^T + \mathbf{R}_k \right]^{-1} \mathbf{K}_k.$$

In the scalar case, these two weighting factors sum to one. In the general case, the weighting factor matrices sum to the identity matrix.

Figure 6.3 illustrates the weighting behavior of the weighting factor for Case 1 as the configuration of the arm and measurement uncertainty change. Only the OWF is shown, as  $PWF = 1 - OWF$  in the scalar case. We see that as the robot nears the singular positions at



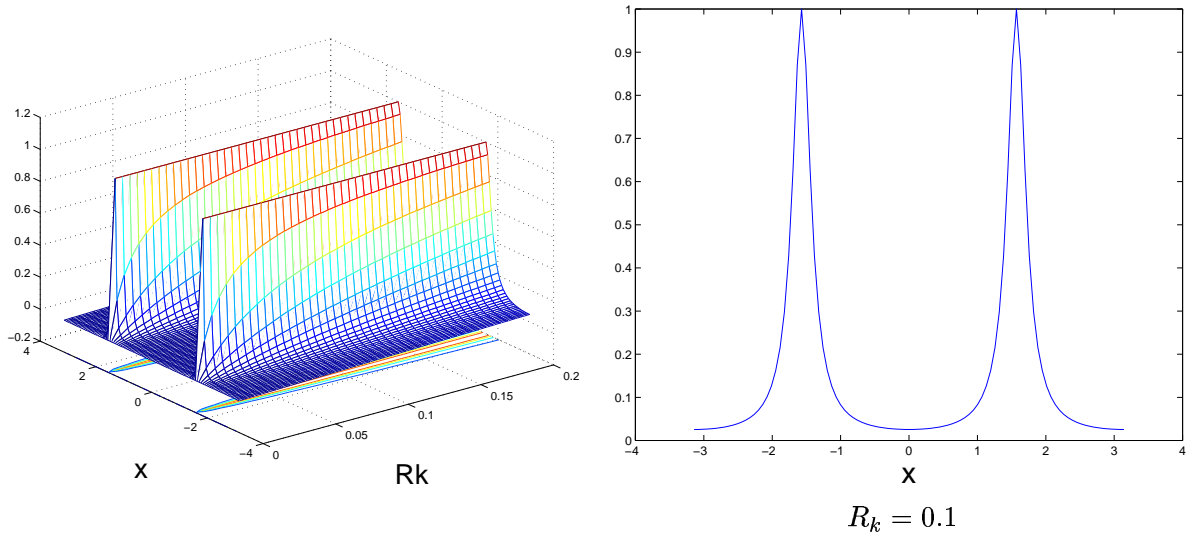
**Figure 6.3** Observation weighting factor for Case 1.

$-\pi/2$  and  $\pi/2$ , the OWF reduces, and the filter favors the predictions over the observations. The extent to which this effect occurs depends on the measurement uncertainty. With very certain measurements ( $R_k$  low) the singularity affects a small portion of the configuration space. As measurements decrease in quality ( $R_k$  rising) the effect spreads to more configurations. A cross section of the curve at  $R_k = 0.1$  is also shown.

### 6.1.2.2 Uncertainty reduction

A metric for the usefulness of observations is the relative uncertainty in the state vector just before and just after a measurement. If an observation is doing a very good job of helping to update the estimate  $\hat{\mathbf{x}}_k$  of the state vector  $\mathbf{x}_k$ , the uncertainty on the estimate just after the observation  $\mathbf{P}_{k,k}$  will be significantly lower than the uncertainty just before the observation  $\mathbf{P}_{k,k-1}$ . Therefore, Figure 6.4 shows the ratio  $P_{k,k}/P_{k,k-1}$  for different configurations of the robot and different observation uncertainties. As in the previous sections, this metric tells us that observations of the end effector near zero radians are very effective at reducing the uncertainty in the state estimate, while observations of the end effector near  $\pi/2$  are not very effective at reducing this uncertainty.

As in the weighting factors, the measurement uncertainty again determines the extent of the singularity on the uncertainty reduction. As  $R_k$  becomes small, little movement away from the singularity is needed for the measurements to become effective again, and reduce the uncertainty in the filter significantly. A cross section of the curve at  $R_k = 0.1$  is also shown.

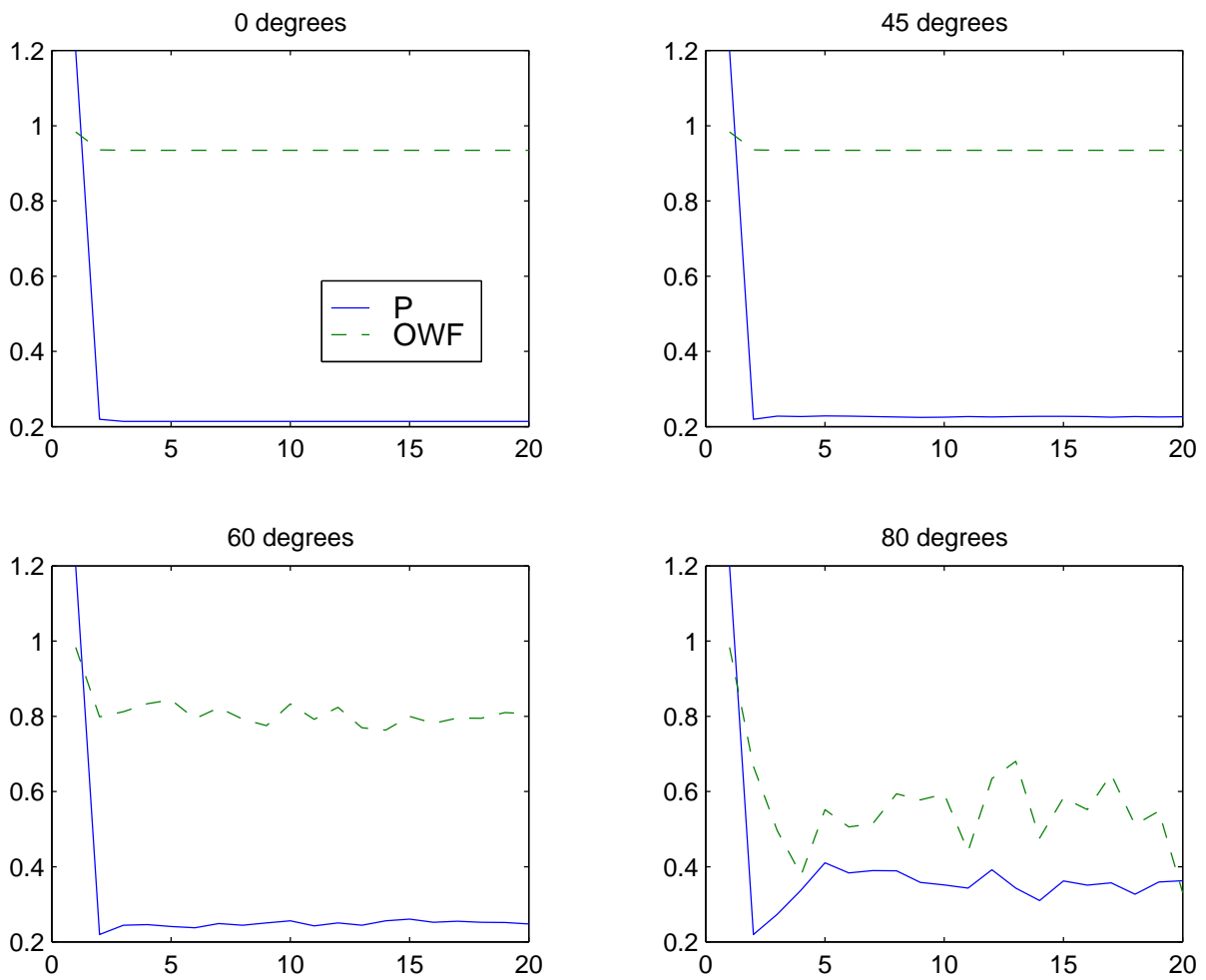


**Figure 6.4**  $\mathbf{P}_{k,k}/\mathbf{P}_{k,k-1}$ : Uncertainty reduction from an observation.

Note that the analysis above is for a single observation. Multiple observations, say at times  $k_1$  and  $k_2$ , would have different uncertainties associated with them,  $\mathbf{R}_{k_1}$  and  $\mathbf{R}_{k_2}$ . This allows the framework to weight different observations of the same feature differently, as is appropriate for their respective  $\mathbf{R}_k$ .

### 6.1.2.3 Convergence of filter

Often, it is of interest to observe the filter components as they change over time. Figure 6.5 shows the variance and weighting components of this filter over time, when the robot is in four different configurations. Of particular interest in this figure are the weights. As the configuration gets closer to the  $\pi/2$  image Jacobian singularity, the weighting favors the previous estimate more than the observations, as we would expect.



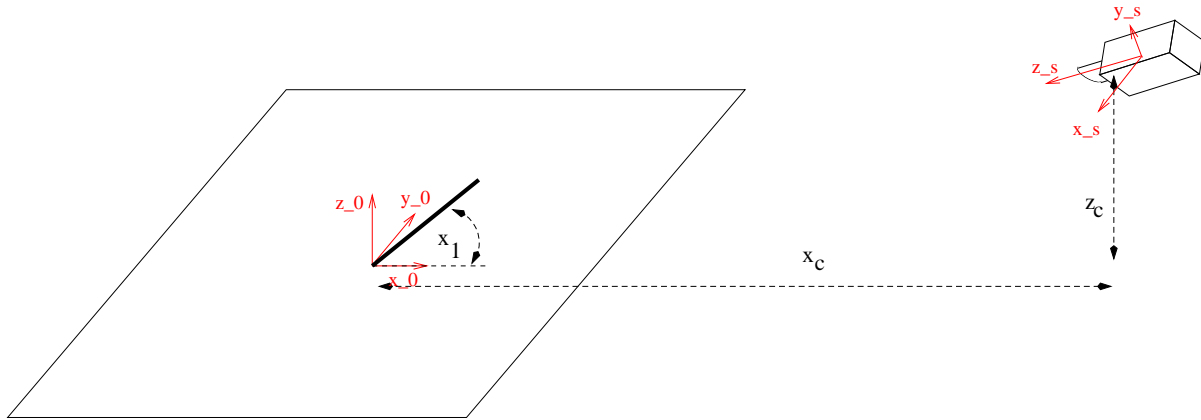
**Figure 6.5** Convergence properties of Case 1.

**Table 6.2** Models used in Case 2.

Object geometric model	1DOF one-link arm
Dynamic model	Constant position
Imaging model	Perspective projection, 2D elevated sensor

## 6.2 Case 2: Elevated 2D Camera

In this section, we present a scenario utilizing the same object model with a slightly more complex imaging model than in Section 6.1, to illustrate the changes that occur with multiple observations. This example, illustrated in Figure 6.6, uses the same arm as the previous case. The models used in this case are shown in Table 6.2. We assume that the robot is embedded in



**Figure 6.6** Case 2.

a 2D workspace and that there is a perspective projection onto a 2D sensor plane (to the right of the robot, at a distance  $x_c$  from the origin, and at an elevation of  $z_c$ ). For convenience, we explicitly define the distance  $d_c = \sqrt{x_c^2 + z_c^2}$ . The sensor plane is assumed to be normal to the ray connecting the center of the sensor to the origin (that is,  $z_s$  in Figure 6.7 points directly at the workspace origin).

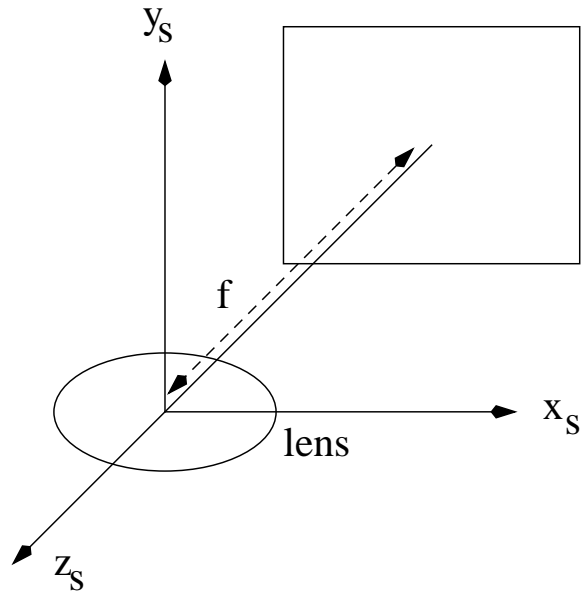


Figure 6.7 Camera coordinate frame.

### 6.2.1 System and filter definition

Using the notation outlined in Section 3.2, the filter for this problem is as follows. The derivation of the equations presented in this section can be found in Appendix B. This system has only one kinematic parameter, so the state vector for this case is again a scalar

$$\mathbf{x}_k = x_1.$$

With a constant position dynamic model, the prediction equations are

$$\begin{aligned} f_k(x_k) &= x_k \\ \frac{\partial f_k}{\partial x_k} &= 1. \end{aligned}$$

Under perspective projection, the image plane location  $(u, v)$  of a point  $(x, y, z)$  in the camera coordinate system taken by a pinhole camera of focal length  $f$  is

$$(u, v) = \left(-\frac{x}{fz}, -\frac{y}{fz}\right).$$



Under this assumption, and assuming a 2D sensor, the observation equations are

$$\mathbf{z}_k = \begin{bmatrix} u_k^1 \\ v_k^1 \end{bmatrix}$$

$$\mathbf{h}_k(\mathbf{x}_k) = \begin{bmatrix} u_k^1(\mathbf{x}_k) \\ v_k^1(\mathbf{x}_k) \end{bmatrix} = \begin{bmatrix} \frac{a_1 \sin(x_k^1)}{f \left( \frac{-x_c(a_1 \cos(x_k^1))}{d_c} + d_c \right)} \\ \frac{z_c a_1 \cos(x_k^1)}{f d_c \left( \frac{-x_c(a_1 \cos(x_k^1))}{d_c} + d_c \right)} \end{bmatrix},$$

where  $(u^1, v^1)$  is defined as the position in the image plane of the distal end of the link. With these definitions,  $\frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k}$ , referred to as the *system Jacobian*, is

$$\frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k} = \begin{bmatrix} \frac{\partial h_k^1}{\partial x_k^1} & \frac{\partial h_k^2}{\partial x_k^1} \end{bmatrix}$$

$$\begin{aligned} \frac{\partial h_k^1}{\partial x_k^1} &= \frac{a_1 \cos(x_k^1)}{f \left( \frac{-x_c(a_1 \cos(x_k^1))}{d_c} + d_c \right)} - \frac{a_1^2 \sin^2(x_k^1) x_c}{f \left( \frac{-x_c(a_1 \cos(x_k^1))}{d_c} + d_c \right)^2 d_c} \\ \frac{\partial h_k^2}{\partial x_k^1} &= -\frac{z_c a_1 \sin(x_k^1)}{f d_c \left( \frac{-x_c(a_1 \cos(x_k^1))}{d_c} + d_c \right)} - \frac{z_c x_c a_1^2 \sin(x_k^1) \cos(x_k^1)}{f d_c^2 \left( \frac{-x_c(a_1 \cos(x_k^1))}{d_c} + d_c \right)^2} \end{aligned}$$

We assume for simplicity that the process noise does not vary with the state,

$$G'_k(x_k) = 1.$$

Under these assumptions, the filter equations are as follows

$$P_{k,k-1} = P_{k-1,k-1} + Q_k$$

$$\hat{x}_{k|k-1} = \hat{x}_{k-1|k-1}$$

$$K_k = P_{k,k-1} \left[ \frac{\partial \mathbf{h}_k}{\partial x_k}(\hat{x}_{k|k-1}) \right]^T \left[ \left[ \frac{\partial \mathbf{h}_k}{\partial x_k}(\hat{x}_{k|k-1}) \right] P_{k,k-1} \left[ \frac{\partial \mathbf{h}_k}{\partial x_k}(\hat{x}_{k|k-1}) \right]^T + \mathbf{R}_k \right]^{-1}$$

$$P_{k,k} = \left[ I - K_k \left[ \frac{\partial \mathbf{h}_k}{\partial x_k}(\hat{x}_{k|k-1}) \right] \right] P_{k,k-1}$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(\mathbf{z}_k - \mathbf{h}_k(\hat{x}_{k|k-1}))$$

## 6.2.2 Filter analysis

We will observe the behavior of the observation weighting matrix and the changes in the covariance matrix after an observation, as the configuration of the arm changes. We are assuming a constant measurement uncertainty for all cases in this section. We begin with a camera at a  $6^\circ$  elevation ( $10z_c = x_c$  in Figure 6.6), then show the changes in the weighting and covariance matrices as the camera is changed to a  $45^\circ$  elevation ( $z_c = x_c$  in Figure 6.6).

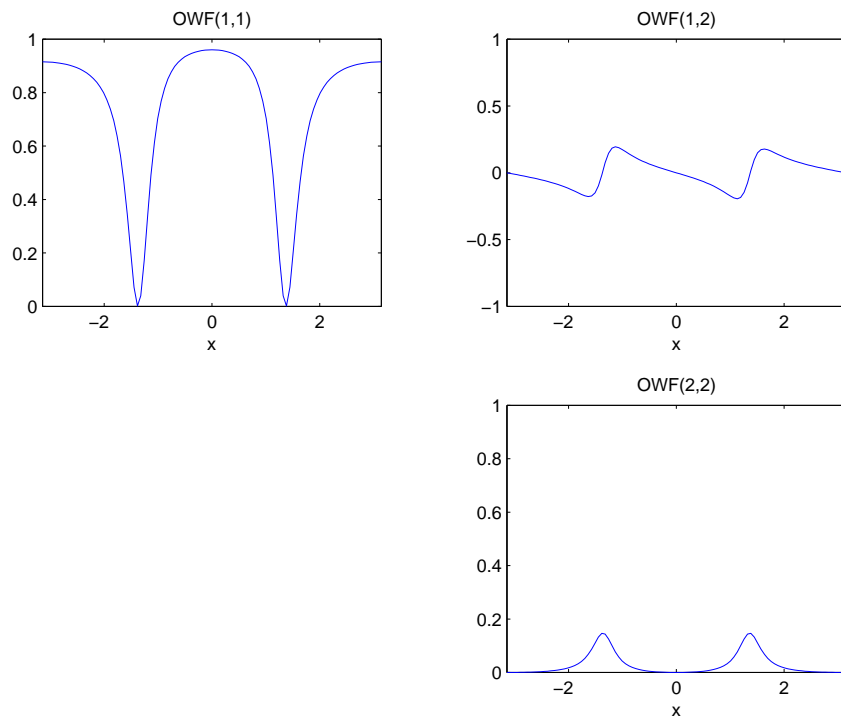
### 6.2.2.1 Weighting of observations and predictions

In the scalar case above, the OWF determined the weight (between zero and unity) assigned to an observation in the filter update. The PWF, equal to  $1 - OWF$  in the scalar case, determined the weight (again between zero and unity) assigned to a prediction in the filter update. The OWF and PWF weighting matrices have the same interpretation in the vector case. The diagonal elements of the OWF determine the weight assigned to an observation when updating the filter's estimate *for that observation*. The update of the state estimate is then determined by inverting the observation function. The off-diagonal elements of the OWF reflect the weights assigned to other observations when updating an observation estimate. The OWF and PWF are diagonal matrices. Thus, each row of the OWF determines the weighting assigned to each observation when updating an element of the filter's observation estimates. Each column of the OWF determines the effect of a single observation on each observation estimate in the filter. Correlations in the observations due to the kinematic and imaging structure of the object tracking scene determine the off-diagonal weight. If the observations are independent (for example, the end effectors of two separate arms) the appropriate elements of the OWF and PWF will be zero.

The weighting of the  $u$  observation in the  $6^\circ$  case (see  $OWF(1,1)$  in Figure 6.8) is similar to the behavior of the single measurement weighting factor in the previous case. The changes are primarily due to the use of perspective projection in this case, instead of orthogonal projection. With the camera in this location, the measurement error in the  $v$  coordinate yields this

measurement largely ineffective in helping to estimate  $x_1$ , due to the low viewing angle. Thus, the weighting assigned to this measurement is fairly low (see  $OWF(2, 2)$  in Figure 6.8).

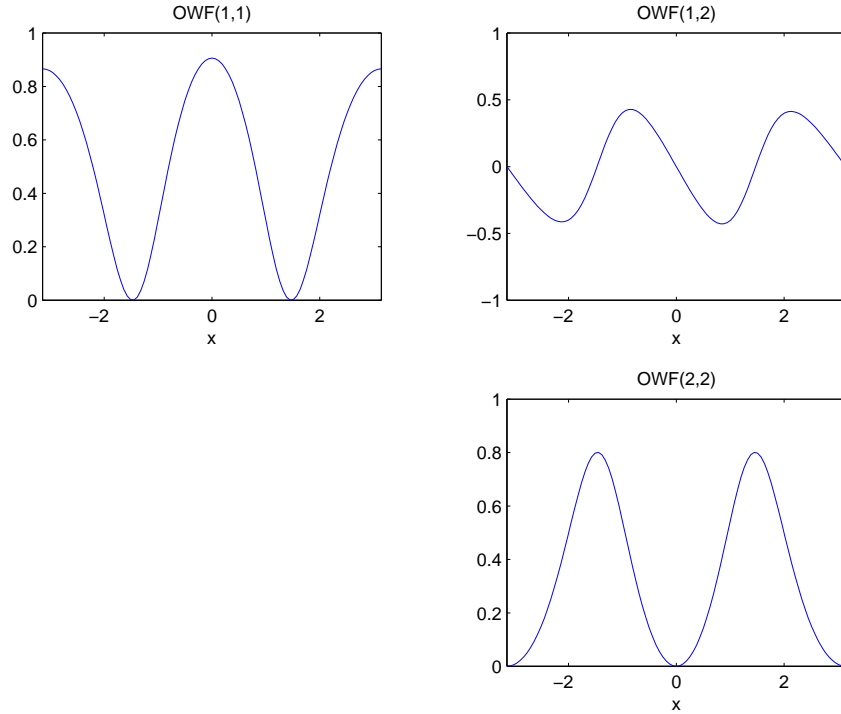
As described above, the weighting factor  $OWF(1, 2)$  determines the usefulness of the  $v$  measurement in updating the prediction for the  $u$  location of the end effector. This is based on the expected correlation between the  $u$  and  $v$  measurements, as determined by the observation model. Likewise, the weighting factor  $OWF(2, 1)$  (not shown, identical to  $OWF(1, 2)$  since the OWF matrix is symmetric) determines the usefulness of the  $u$  measurement in updating the prediction for the  $v$  location of the end effector. Since the correlation between observations can be negative or positive, the off-diagonal elements of the OWF are not restricted to the zero-one range, but the full negative one-positive one range.



**Figure 6.8** Observation weighting factor matrix for Case 2,  $6^\circ$  elevation.

When the viewing angle is changed to  $45^\circ$ , the  $v$  observation becomes more important, and is used more to help estimate  $x_1$ . The weight  $OWF(2, 2)$  in Figure 6.9 now ranges between

zero and 0.8. Due to the observation model,  $OWF(1,1)$  is low when  $OWF(2,2)$  is high and vice versa.

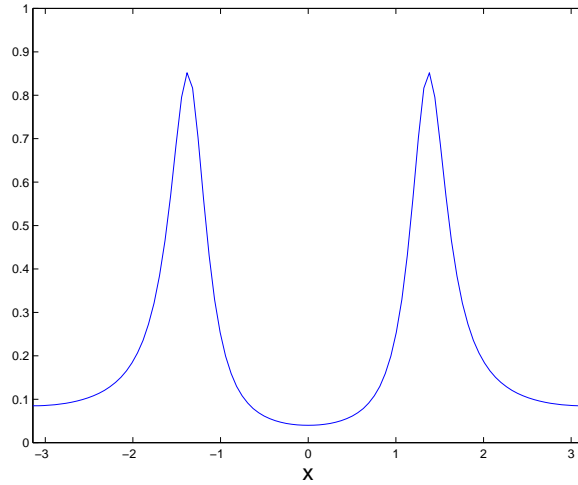


**Figure 6.9** Observation weighting factor matrix for Case 2, 45° elevation.

### 6.2.2.2 Uncertainty reduction

In this section, we investigate the reduction in uncertainty due to a single observation. We investigate  $\mathbf{P}_{k,k}/\mathbf{P}_{k,k-1}$ , the ratio of the uncertainty in the state estimate just before an observation to the uncertainty just after the observation. Again, the 6° case closely resembles the 1D sensor case presented in Section 6.1. Near the singularities in the observation function shown in Figure 6.10 for the  $u$  observation at  $x_1 = -\pi/2$  and  $x_1 = \pi/2$ , the uncertainty does not decrease much due to a single observation.

When the camera is raised to a 45° elevation the observations of the  $v$  coordinate of the end effector begin to help. Since the  $v$  coordinate *does* change with changes in  $x_1$  near  $x_1 = -\pi/2$  and  $x_1 = \pi/2$ ,  $v$  observations in these regions are used to update the estimate for  $x_1$ ,



**Figure 6.10**  $\mathbf{P}_{k,k}/\mathbf{P}_{k,k-1}$ : Uncertainty reduction from an observation. Case 2,  $6^\circ$  elevation.

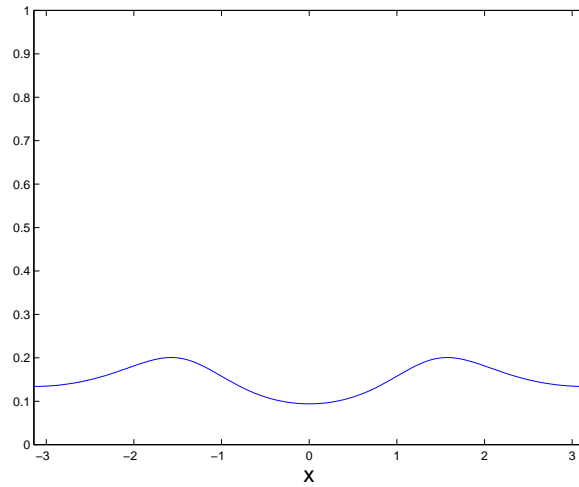
**Table 6.3** Models used in Case 3.

Object geometric model	Two-link arm
Dynamic model	Constant position
Imaging model	Perspective projection

where in previous cases only the observation prediction was available. The presence of reliable observations in these regions is evident in Figure 6.11, where there is significant reduction in uncertainty over the entire configuration space. With multiple observations singular points in one observation function can be tolerated, provided the state elements being estimated by the singular observation are observable (in the dynamic systems sense, see Section 4.3 for discussion) from the nonsingular observations.

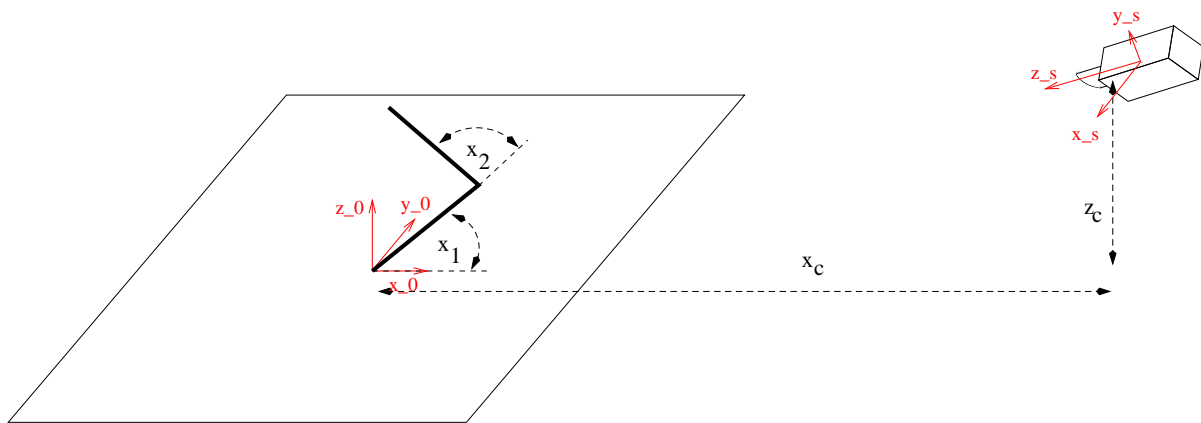
### 6.3 Case 3: More Complex Models

In this section, we present a scenario utilizing a more complex object and imaging model than in Section 6.1, to further illustrate the use of the framework for tracking. This example, illustrated in Figure 6.12, uses a two-link planar arm. The models used in this case are shown in Table 6.3. The configuration space for this robot is the standard 2D space with dimensions

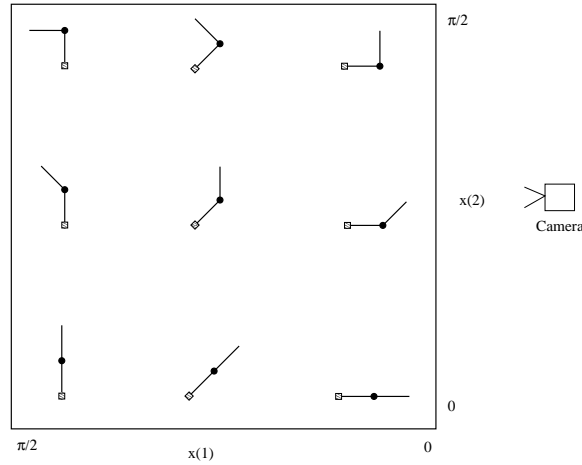


**Figure 6.11**  $\mathbf{P}_{k,k}/\mathbf{P}_{k,k-1}$ : Uncertainty reduction from an observation. Case 2,  $45^\circ$  elevation.

consisting of the angles  $(x_1, x_2)$  as depicted in the figure. This configuration space is illustrated in Figure 6.13. This figure shows a few representative configurations of the arm, as viewed from directly above the workspace. Due to the increased amount of complexity, this section will present only the portion of configuration space from  $(0, 0)$  to  $(\pi/2, \pi/2)$ . The remaining portion of configuration space exhibits similar behavior, with analogous singularities. We assume that the robot is embedded in a 2D workspace and that there is a perspective projection onto a 2D



**Figure 6.12** Case 3.



**Figure 6.13** Configuration space for Case 3.

sensor plane (to the right of the robot, at a distance  $x_c$  from the origin, and at an elevation of  $z_c$ ). The imaging model is the same as for Case 2 above.

### 6.3.1 System and filter definition

Using the notation outlined in Section 3.2, the filter for this problem is as follows. The derivation of the equations presented in this section can be found in Appendix C. This system has two kinematic parameters, so the state vector for this case has two entries

$$\mathbf{x}_k = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}.$$

With a constant position dynamic model, the prediction equations are

$$\begin{aligned} \mathbf{f}_k(\mathbf{x}_k) &= \mathbf{x}_k \\ \frac{\partial \mathbf{f}_k}{\partial \mathbf{x}_k} &= I. \end{aligned}$$

Under perspective projection, the image plane location  $(u, v)$  of a point  $(x, y, z)$  in the camera coordinate system taken by a pinhole camera of focal length  $f$  is

$$(u, v) = \left(-\frac{x}{fz}, -\frac{y}{fz}\right).$$

Under this assumption, and assuming a 2D sensor, the observation equations are

$$\mathbf{z}_k = \begin{bmatrix} u_k^1 \\ v_k^1 \\ u_k^2 \\ v_k^2 \end{bmatrix}$$

$$\mathbf{h}_k(\mathbf{x}_k) = \begin{bmatrix} u_k^1(\mathbf{x}_k) \\ v_k^1(\mathbf{x}_k) \\ u_k^2(\mathbf{x}_k) \\ v_k^2(\mathbf{x}_k) \end{bmatrix} = \begin{bmatrix} \frac{a_1 \sin(x_k^1)}{f \left( \frac{-x_c(a_1 \cos(x_k^1))}{d_c} + d_c \right)} \\ \frac{z_c a_1 \cos(x_k^1)}{f d_c \left( \frac{-x_c(a_1 \cos(x_k^1))}{d_c} + d_c \right)} \\ \frac{a_1 \sin(x_k^1) + a_2 \sin(x_k^1 + x_k^2)}{f \left( \frac{-x_c(a_1 \cos(x_k^1) + a_2 \cos(x_k^1 + x_k^2))}{d_c} + d_c \right)} \\ \frac{z_c(a_1 \cos(x_k^1) + a_2 \cos(x_k^1 + x_k^2))}{f d_c \left( \frac{-x_c(a_1 \cos(x_k^1) + a_2 \cos(x_k^1 + x_k^2))}{d_c} + d_c \right)} \end{bmatrix},$$

where  $(u^1, v^1)$  and  $(u^2, v^2)$  are defined as the position in the image plane of the distal ends of the first and second links, respectively. With these definitions,  $\frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k}$ , referred to as the *system*



Jacobian, is

$$\frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k} = \begin{bmatrix} \frac{\partial h_k^1}{\partial x_k^1} & \frac{\partial h_k^1}{\partial x_k^2} \\ \frac{\partial h_k^2}{\partial x_k^1} & \frac{\partial h_k^2}{\partial x_k^2} \\ \frac{\partial h_k^3}{\partial x_k^1} & \frac{\partial h_k^3}{\partial x_k^2} \\ \frac{\partial h_k^4}{\partial x_k^1} & \frac{\partial h_k^4}{\partial x_k^2} \end{bmatrix}$$

$$\frac{\partial h_k^1}{\partial x_k^1} = \frac{a_1 \cos(x_k^1)}{f \left( \frac{-x_c(a_1 \cos(x_k^1))}{d_c} + d_c \right)} - \frac{a_1^2 \sin^2(x_k^1) x_c}{f \left( \frac{-x_c(a_1 \cos(x_k^1))}{d_c} + d_c \right)^2 d_c}$$

$$\frac{\partial h_k^1}{\partial x_k^2} = 0$$

$$\frac{\partial h_k^2}{\partial x_k^1} = -\frac{z_c a_1 \sin(x_k^1)}{f d_c \left( \frac{-x_c(a_1 \cos(x_k^1))}{d_c} + d_c \right)} - \frac{z_c x_c a_1^2 \sin(x_k^1) \cos(x_k^1)}{f d_c^2 \left( \frac{-x_c(a_1 \cos(x_k^1))}{d_c} + d_c \right)^2}$$

$$\frac{\partial h_k^2}{\partial x_k^2} = 0$$

$$\begin{aligned} \frac{\partial h_k^3}{\partial x_k^1} &= -\frac{a_1 \cos(x_k^1) + a_2 \cos(x_k^1 + x_k^2)}{f \left( \frac{-x_c(a_1 \cos(x_k^1) + a_2 \cos(x_k^1 + x_k^2))}{d_c} + d_c \right)} \\ &\quad - \frac{(a_1 \cos(x_k^1) + a_2 \cos(x_k^1 + x_k^2))^2 x_c}{f \left( \frac{-x_c(a_1 \cos(x_k^1) + a_2 \cos(x_k^1 + x_k^2))}{d_c} + d_c \right)^2 d_c} \end{aligned}$$

$$\begin{aligned} \frac{\partial h_k^3}{\partial x_k^2} &= \frac{a_2 \cos(x_k^1 + x_k^2)}{f \left( \frac{-x_c(a_1 \cos(x_k^1) + a_2 \cos(x_k^1 + x_k^2))}{d_c} + d_c \right)} \\ &\quad - \frac{a_1 \sin(x_k^1) + a_2 \sin(x_k^1 + x_k^2) x_c a_2 \sin(x_k^1 + x_k^2)}{f \left( \frac{-x_c(a_1 \cos(x_k^1) + a_2 \cos(x_k^1 + x_k^2))}{d_c} + d_c \right)^2 d_c} \end{aligned}$$

$$\begin{aligned}\frac{\partial h_k^4}{\partial x_k^1} &= -\frac{z_c(a_1 \sin(x_k^1) + a_2 \sin(x_k^1 + x_k^2))}{f d_c \left( \frac{-x_c(a_1 \cos(x_k^1) + a_2 \cos(x_k^1 + x_k^2))}{d_c} + d_c \right)} \\ &\quad - \frac{z_c(a_1 \cos(x_k^1) + a_2 \cos(x_k^1 + x_k^2)) x_c (a_1 \sin(x_k^1) + a_2 \sin(x_k^1 + x_k^2))}{f d_c^2 \left( \frac{-x_c(a_1 \cos(x_k^1) + a_2 \cos(x_k^1 + x_k^2))}{d_c} + d_c \right)^2} \\ \frac{\partial h_k^4}{\partial x_k^2} &= -\frac{z_c(a_2 \sin(x_k^1 + x_k^2))}{f d_c \left( \frac{-x_c(a_1 \cos(x_k^1) + a_2 \cos(x_k^1 + x_k^2))}{d_c} + d_c \right)} \\ &\quad - \frac{z_c(a_1 \cos(x_k^1) + a_2 \cos(x_k^1 + x_k^2)) x_c a_2 \sin(x_k^1 + x_k^2)}{f d_c^2 \left( \frac{-x_c(a_1 \cos(x_k^1) + a_2 \cos(x_k^1 + x_k^2))}{d_c} + d_c \right)^2}\end{aligned}$$

We assume for simplicity that the process noise does not vary with the state,

$$\mathbf{G}'_k(\mathbf{x}_k) = \mathbf{I}.$$

Under these assumptions, the filter equations are as follows

$$\mathbf{P}_{k,k-1} = \mathbf{P}_{k-1,k-1} + \mathbf{Q}_k$$

$$\hat{\mathbf{x}}_{k|k-1} = \hat{\mathbf{x}}_{k-1|k-1}$$

$$\mathbf{K}_k = \mathbf{P}_{k,k-1} \left[ \frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k}(\hat{\mathbf{x}}_{k|k-1}) \right]^T \left[ \left[ \frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k}(\hat{\mathbf{x}}_{k|k-1}) \right] \mathbf{P}_{k,k-1} \left[ \frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k}(\hat{\mathbf{x}}_{k|k-1}) \right]^T + \mathbf{R}_k \right]^{-1}$$

$$\mathbf{P}_{k,k} = \left[ \mathbf{I} - \mathbf{K}_k \left[ \frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k}(\hat{\mathbf{x}}_{k|k-1}) \right] \right] \mathbf{P}_{k,k-1}$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\mathbf{z}_k - \mathbf{h}_k(\hat{\mathbf{x}}_{k|k-1}))$$

### 6.3.2 Filter analysis

As in the previous analysis, we will be looking at the OWF weighting matrices and the uncertainty reduction from a single observation as the configuration of the arm changes. In this case, we are assuming a constant measurement uncertainty in all cases. We begin with a camera at a 45° elevation ( $z_c = x_c$  in Figure 6.12), then show the changes in the weighting matrix as the camera is changed to a 6° elevation ( $10z_c = x_c$  in Figure 6.12).

### 6.3.2.1 Weighting of predictions and observations

We first turn our attention to the weighting matrices for the problem. In Figure 6.14 we illustrate the OWF weighting factor matrix for the  $45^\circ$  elevation case as the configuration changes. Recall that the diagonal components in this matrix reflect the effect of each observa-

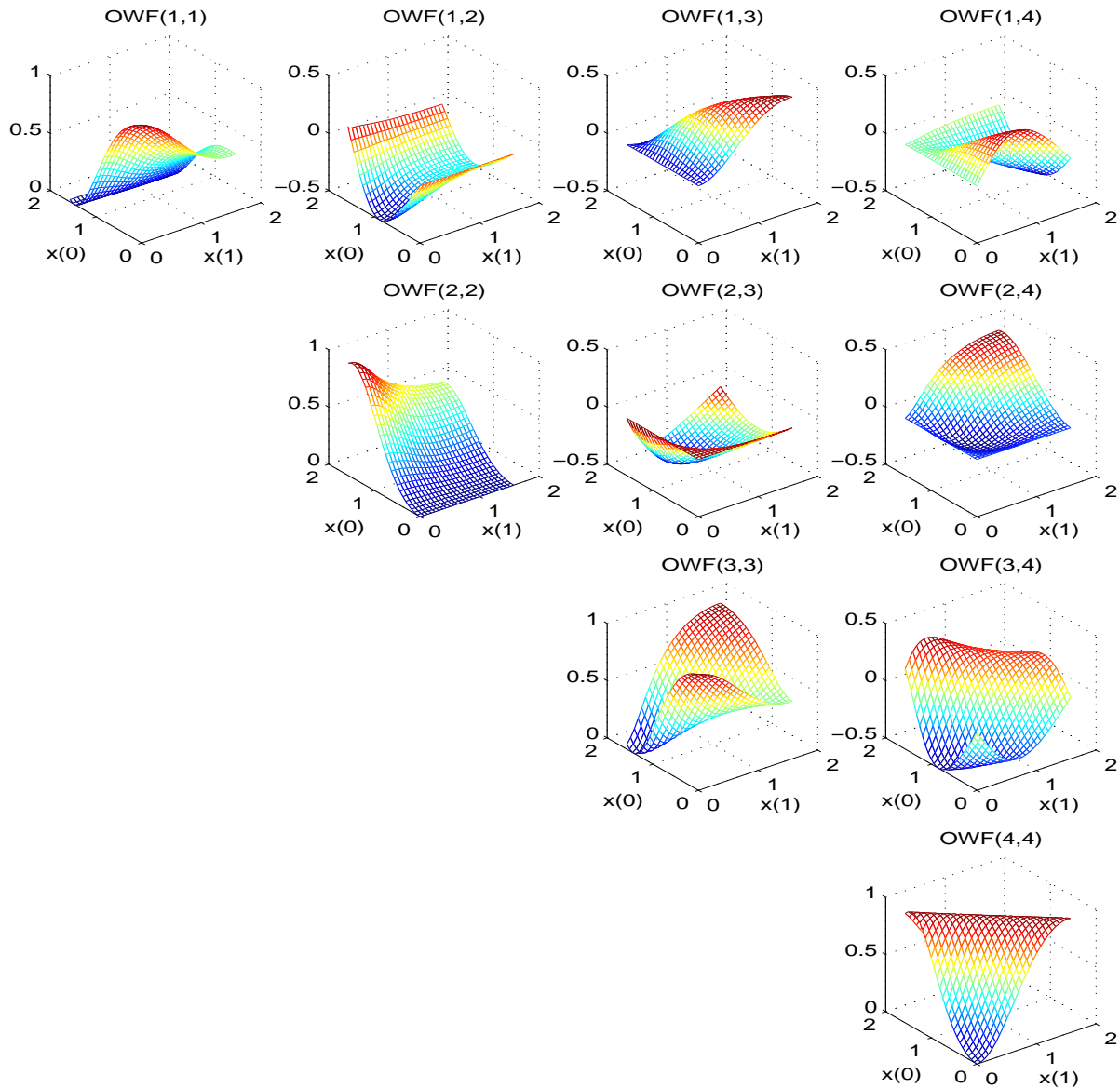


Figure 6.14 Observation weighting factor matrix for  $45^\circ$  camera elevation.

tion in the update equations, and the off-diagonal components reflect the correlation between observations.

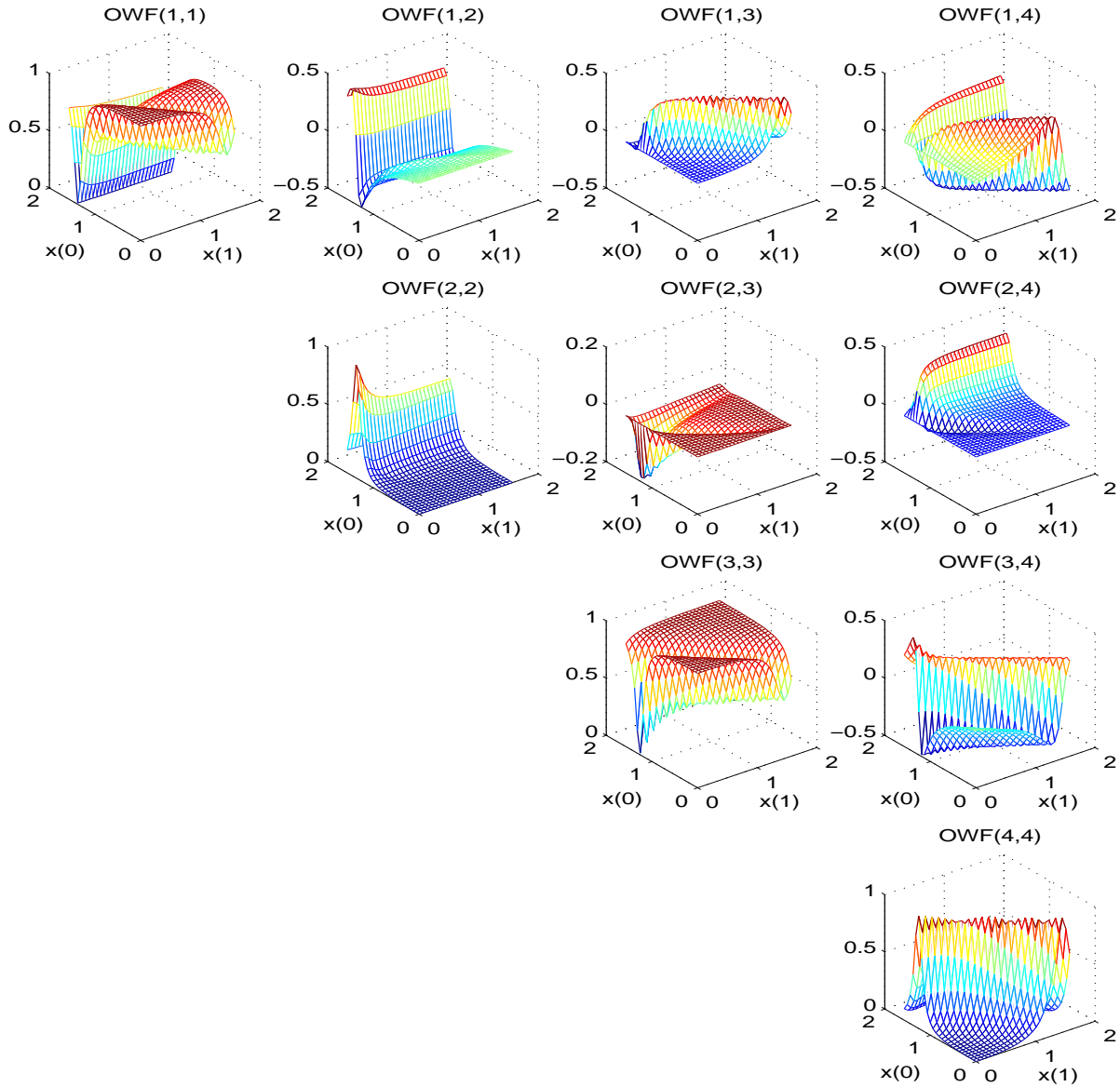
The weighting near the  $(0, 0)$  corner of the illustrated portion of configuration space is most similar to the previous cases. Figure 6.13 shows that in this configuration, the arm is straight and pointed directly at the camera. The  $u$  observations for each link are used to update the filter estimates ( $OWF(1, 1)$  and  $OWF(3, 3)$ ), and the  $v$  observations are ignored ( $OWF(2, 2)$  and  $OWF(4, 4)$ ). Analysis of the off-diagonal elements reveals that there is significant correlation between the  $u$  and  $v$  coordinate of each link (between  $u_1$  and  $v_1$ , and between  $u_2$  and  $v_2$ ), but not between observations from different links.

Next we investigate the changes that occur in the  $OWF$  when the camera is moved to a  $6^\circ$  elevation. The biggest change in the  $OWF$  is the decreased usefulness of the  $v$  coordinates of the elbow and end effector of the arm seen in Figure 6.15  $OWF(2, 2)$  and  $OWF(4, 4)$ . Only in configurations where the appropriate link is almost exactly parallel to the image plane are the  $v$  observations used. This occurs near  $x_0 = \pi/2$  for the first link and along a line between  $(\pi/2, 0)$  and  $(0, \pi/2)$  for the second link.

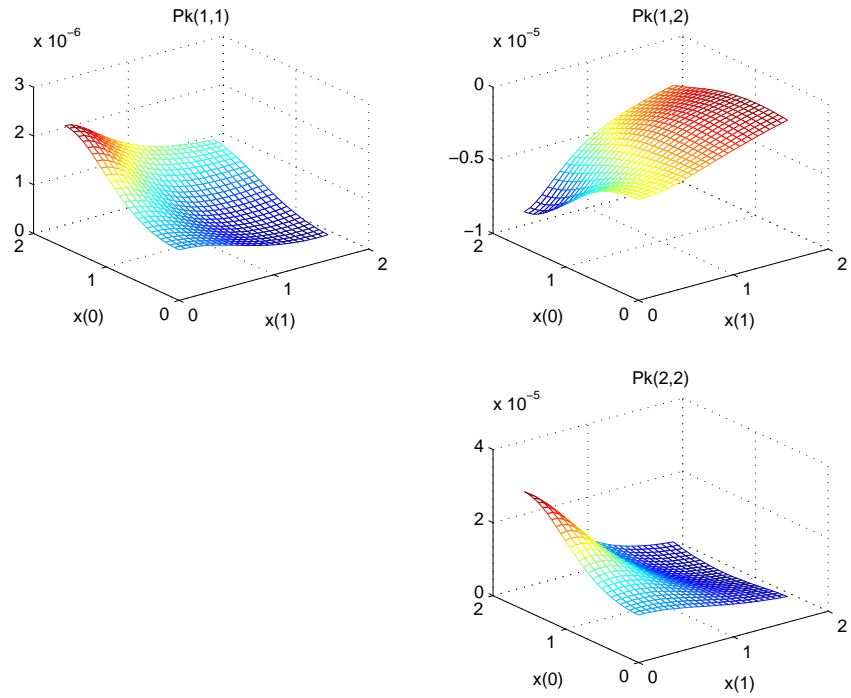
### 6.3.2.2 Uncertainty reduction

Recalling the structure of a covariance matrix Equation (4.2), we now observe in Figure 6.16 the changes in  $\mathbf{P}_k$  due to a single observation. The *a priori* covariance matrix in this case was the identity matrix, so this case is analogous to our analysis of the  $P_{k,k}/P_{k,k-1}$  ratio in Case 1. Recall that an observation in this case includes measurements of the image plane locations of the elbow and end effector of the two-link arm.

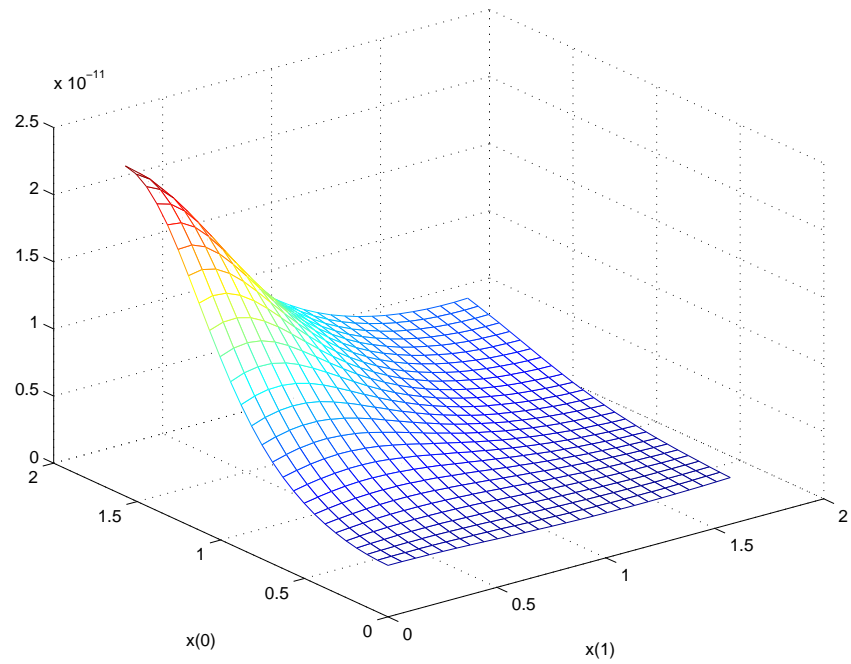
A summary of the uncertainty reduction from a single observation can be seen in Figures 6.16 and 6.17. Recall that  $|\mathbf{P}_k|$  is proportional to the volume of the error ellipsoid in the  $x_0, x_1$  plane.



**Figure 6.15** Observation weighting factor matrix for  $6^\circ$  camera elevation.

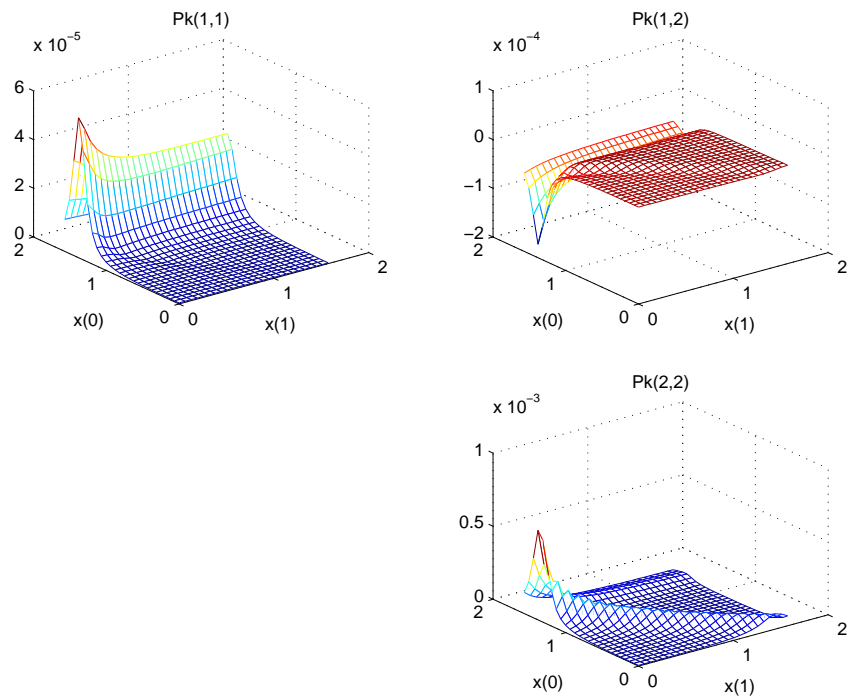


(a) Components of  $\mathbf{P}_k$

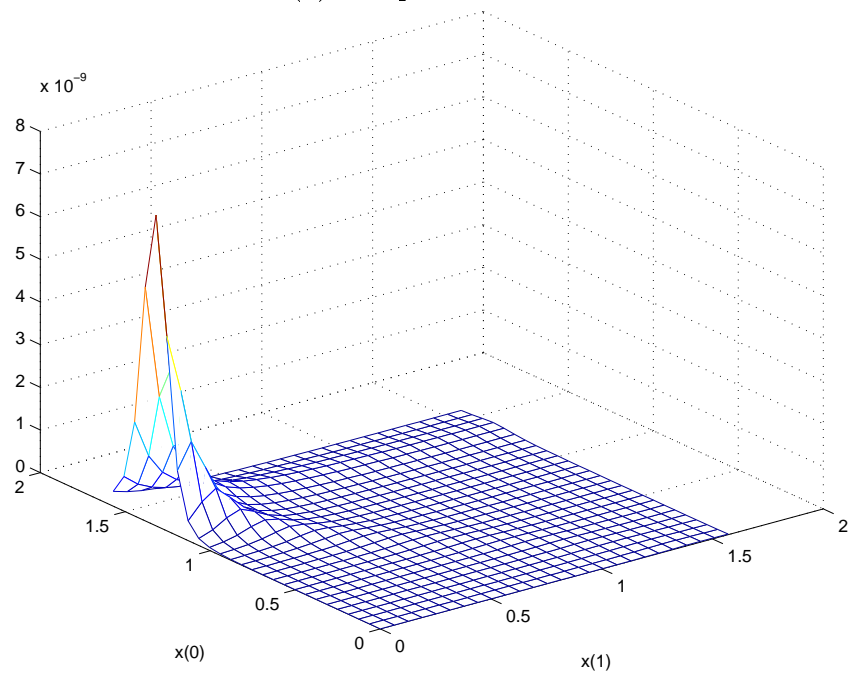


(b) Determinant of  $\mathbf{P}_k$

**Figure 6.16** Error covariance matrix for  $45^\circ$  camera elevation.



(a) Components of  $\mathbf{P}_k$



(b) Determinant of  $\mathbf{P}_k$

**Figure 6.17** Error covariance matrix for 6° camera elevation.

**Table 6.4** Models used in Case 4.

Object geometric model	1DOF one-link arm
Dynamic model	Constant joint space velocity
Imaging model	Orthogonal projection

The variance of  $x_0$ ,  $\mathbf{P}_k^{1,1}$ , has a minimum near  $(0, \pi/2)$ . In this configuration, the first link is pointing at the camera, and the second link is directly orthogonal to the first link. Thus, the  $u$  coordinate of each link gives good information about the value of  $x_0$ . The maximum for  $\mathbf{P}_k^{1,1}$  is at  $(\pi/2, 0)$ , when the arm is straight and pointing directly orthogonal to the camera.

While the situation in this section is more complex than that presented earlier, the results are still intuitively satisfying and have a plausible geometric interpretation. This example is even more useful, since it shows the interactions between different links of the arm, and how these interactions change with configuration. The framework allows us to capture these interactions, and use the structure given by the model of the arm, to intelligently interpret the usefulness of observations.

## 6.4 Case 4: Use of Dynamic Model

In this section, we present the implementation changes required to utilize a dynamic model for the scenario described in Section 6.1. In this section, we estimate the value and first time derivative of  $\theta$ . The derivation of the equations presented in this section can be found in Appendix D. The models used in this case are shown in Table 6.4. Hereafter, superscripts will be used to denote vector or matrix indices, and subscripts will continue to be used to denote time indices.

$$\mathbf{x}_k = \begin{bmatrix} x_k^1 \\ x_k^2 \end{bmatrix} = \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix}.$$

With a dynamic model, the estimate at time  $k$  Equation (6.2) is no longer the previous estimate. Now, we can adjust by the estimated velocity (multiplied by the elapsed time  $\Delta$ ), to



give a presumably more accurate extrapolated estimate

$$\mathbf{f}_k(\mathbf{x}_k) = \begin{bmatrix} x_k^1 + \Delta x_k^2 \\ x_k^2 \end{bmatrix}$$

$$\frac{\partial \mathbf{f}_k}{\partial \mathbf{x}_k} = \begin{bmatrix} 1 & \Delta \\ 0 & 1 \end{bmatrix}.$$

In the observation function, we preserve most of Equation (6.4), adding a row to denote that we cannot observe the velocity:

$$\mathbf{h}_k(\mathbf{x}_k) = \begin{bmatrix} a_1 \sin(x_k) \\ 0 \end{bmatrix}$$

$$\frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k} = \begin{bmatrix} a_1 \cos(x_k) \\ 0 \end{bmatrix}$$

We again assume for simplicity that the process noise does not vary with the state,

$$G'_k(\mathbf{x}_k) \equiv I.$$

Under these assumptions, the filter equations are as follows

$$\mathbf{P}_{k,k-1} = \begin{bmatrix} 1 & \Delta \\ 0 & 1 \end{bmatrix} \mathbf{P}_{k-1,k-1} \begin{bmatrix} 1 & 0 \\ \Delta & 1 \end{bmatrix} + \mathbf{Q}_k \quad (6.9)$$

$$= \begin{bmatrix} \mathbf{P}_{k-1,k-1}^{1,1} + 2\Delta \mathbf{P}_{k-1,k-1}^{1,2} + \Delta^2 \mathbf{P}_{k-1,k-1}^{2,2} & \mathbf{P}_{k-1,k-1}^{1,2} + \Delta \mathbf{P}_{k-1,k-1}^{2,2} \\ \mathbf{P}_{k-1,k-1}^{1,2} + \Delta \mathbf{P}_{k-1,k-1}^{2,2} & \mathbf{P}_{k-1,k-1}^{2,2} \end{bmatrix} + \mathbf{Q}_k$$

$$\hat{\mathbf{x}}_{k|k-1} = \begin{bmatrix} \hat{x}_{k-1|k-1}^1 + \hat{x}_{k-1|k-1}^2 \Delta \\ \hat{x}_{k-1|k-1}^2 \end{bmatrix} \quad (6.10)$$

$$\mathbf{K}_k = \frac{a_1 \cos(\hat{\mathbf{x}}_{k-1})}{a_1^2 \cos^2(\hat{\mathbf{x}}_{k-1}) \mathbf{P}_{k,k-1}^{1,1} + \mathbf{R}_k} \begin{bmatrix} \mathbf{P}_{k,k-1}^{1,1} \\ \mathbf{P}_{k,k-1}^{1,2} \end{bmatrix} \quad (6.11)$$

$$\mathbf{P}_{k,k} = \begin{bmatrix} 1 - a_1 \cos(\mathbf{K}_k^1) & 0 \\ -a_1 \cos(\mathbf{K}_k^2) & 1 \end{bmatrix} \mathbf{P}_{k,k-1} \quad (6.12)$$

**Table 6.5** Models used in Case 5.

Object geometric model	Three-link arm (PUMA kinematics)
Dynamic model	Constant joint space velocity
Imaging model	Perspective projection

Note that the above equations are essentially matrix versions of Equations (6.5)-(6.8), with additional terms involving increased uncertainty, and increased predictive power, due to elapsed time and the dynamic model. Also of note is the gain matrix  $\mathbf{K}_k$  in Equation (6.11), which updates the velocity estimate based upon a fraction of the prediction correction.

## 6.5 Case 5: 3DOF PUMA Robotic Arm

In this section, we present a tracking scenario driving this research, that of a three degree of freedom robotic arm under perspective projection. In this case, the three degrees of freedom used to position the tool at any position in the workspace are tracked. The final three degrees of freedom determine the orientation of the tool about this point in space. The form of the filter is the same as described above, but the components are more complicated. The position of the camera in the world coordinate system is known and fixed, given by a homogeneous transformation matrix *Camera*. The models used in this case are shown in Table 6.5.

### 6.5.1 System and filter definition

The filter for this example is a straightforward extension to that described above. The derivation of the equations presented in this section can be found in Appendix E. The state vector contains the joint angles and velocities

$$\mathbf{x}_k = [q_1 \dots q_3 \dot{q}_1 \dots \dot{q}_3]^T,$$

and the dynamic model is as described in Section 6.4

$$\mathbf{f}_k(\mathbf{x}_k) = \begin{bmatrix} \mathbf{q}_k + \Delta \dot{\mathbf{q}}_k \\ \dot{\mathbf{q}}_k \end{bmatrix}$$

$$\frac{\partial \mathbf{f}_k}{\partial \mathbf{x}_k} = \begin{bmatrix} \mathbf{I}_3 & \Delta \mathbf{I}_3 \\ 0\mathbf{I}_3 & \mathbf{I}_3 \end{bmatrix},$$

where  $\mathbf{I}_3$  is a  $3 \times 3$  identity matrix.

Because many more features are required to track an arm with three degrees of freedom, the observation function becomes more complex. We will introduce some additional notation to simplify the function. A feature is described by a vector  $\mathbf{p}$ , specifying the location of the feature, and a number, specifying the link to which the coordinate system containing  $\mathbf{p}$  is specified. We will describe a point  $\mathbf{p}$  in some coordinate frame by a four-vector  $\mathbf{p} = [x \ y \ z \ 1]$ , where  $x$ ,  $y$ , and  $z$  are the coordinates of the point. A transformation from the coordinate frame rigidly attached to link  $i$  into that rigidly attached to link  $i + 1$  will be given by the standard  $4 \times 4$  Denevit-Hartenberg homogeneous transformation matrix [93]  $\mathbf{A}_i$ . We define the function  $\mathbf{T}(f)$  to be the cascade of the individual  $\mathbf{A}_i$  matrices from the coordinate system of feature  $f$  to the world coordinate system. For example, if feature  $f$  is specified in the coordinate system attached to link 4,  $\mathbf{T}(f) = \mathbf{A}_1\mathbf{A}_2\mathbf{A}_3\mathbf{A}_4$ . The function `PERSPECTIVE()` returns the perspective projection of the point specified.

$$\mathbf{h}_k(\mathbf{x}_k) = \begin{bmatrix} \text{PERSPECTIVE}(\text{Camera } A(1)\mathbf{p})^T \\ \text{PERSPECTIVE}(\text{Camera } A(2)\mathbf{p})^T \\ \vdots \\ \text{PERSPECTIVE}(\text{Camera } A(F-1)\mathbf{p})^T \\ \text{PERSPECTIVE}(\text{Camera } A(F)\mathbf{p})^T \end{bmatrix},$$

where there are  $F$  features to be observed. Note that  $\mathbf{h}()$  is a column vector, and contains the  $(u, v)$  coordinates of each feature. Also note that all  $\mathbf{A}_i$  and thus  $\mathbf{T}()$  are dependent on  $\mathbf{q}$ . With a fully specified geometric model, however, closed form solutions for  $\mathbf{h}(\mathbf{x}_k)$  and  $\frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k}(\mathbf{x}_k)$  can be

**Table 6.6** Models used in Case 6.

Object geometric model	Two-link arm, unknown link lengths
Dynamic model	Constant joint space velocity
Imaging model	Perspective projection

found. We again assume for simplicity that the process noise does not vary with the state,

$$G'_k(\mathbf{x}_k) \equiv I.$$

## 6.6 Case 6: Incomplete Object Model

This section describes the application of a tracking system to a two-link robotic arm with two degrees of freedom under perspective projection, where the lengths of the two links will be estimated along with the configuration parameters. The models used in this case are shown in Table 6.6. A simple augmentation of  $\mathbf{x}_k$  is all that is required to use the system in this case. The derivation of the equations presented in this section can be found in Appendix F.

### 6.6.1 System and filter definition

The state vector in this case contains both the joint angles  $\mathbf{q}$  of the robot, the velocities of those angles, and the link lengths  $\mathbf{a}$  of the robot,  $\mathbf{x}_k = [q_1 \ q_2 \ \dot{q}_1 \ \dot{q}_2 \ a_1 \ a_2]^T$ . Note that the velocities of the lengths are not estimated, as we assume that the lengths are unknown constants.

The dynamic model incorporating these assumptions is

$$\mathbf{f}_k(\mathbf{x}_k) = \begin{bmatrix} \mathbf{q}_1 + \Delta\dot{\mathbf{q}}_1 \\ \mathbf{q}_2 + \Delta\dot{\mathbf{q}}_2 \\ \dot{q}_1 \\ \dot{q}_2 \\ a_1 \\ a_2 \end{bmatrix},$$

and the observation function remains the same as in Section 6.3 with the exception that  $a_1$  and  $a_2$  are now variable:

$$\mathbf{h}_k(\mathbf{x}_k) = \begin{bmatrix} u_k^1(\mathbf{x}_k) \\ v_k^1(\mathbf{x}_k) \\ u_k^2(\mathbf{x}_k) \\ v_k^2(\mathbf{x}_k) \end{bmatrix}.$$

## 6.7 Extensions to Other Situations

Extending the framework to new situations requires defining assumptions about the object geometric model and the imaging model, to describe an observation function. In addition, assumptions about the dynamic model (which may dictate augmenting the state vector) must be described.

Once closed forms have been described for the observation function  $\mathbf{h}_k(\mathbf{x}_k)$  and the dynamic model  $\mathbf{f}_k(\mathbf{x}_k)$ , the relevant partial derivatives  $\frac{\partial h_k}{\partial x_k}(\mathbf{x}_k)$  and  $\frac{\partial f_k}{\partial x_k}(\mathbf{x}_k)$  must be derived, so that the given form of the EKF can be directly implemented. Since this linearization can be done algebraically, the system need not estimate the derivatives from the observed data, a process that is well known to be sensitive to noise in the observations.

## CHAPTER 7

### RESULTS

In this chapter we illustrate the effectiveness of the tracking system in several situations. Again, each situation illustrates a particular feature of the tracking system. We begin by showing tracking results for an arm with three degrees of freedom in several situations, and we use this case to illustrate the features of the system. We then show results for an arm with two degrees of freedom and an incomplete geometric model.

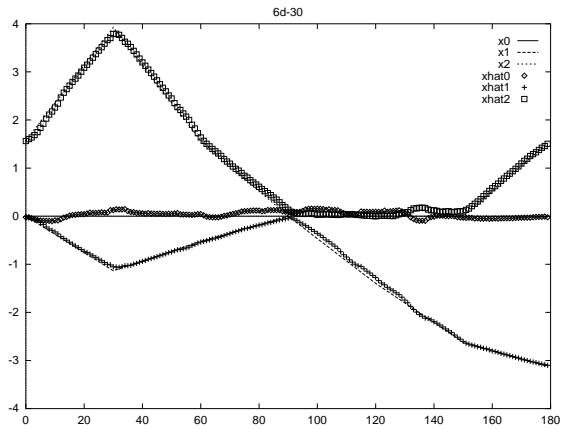
#### 7.1 Case 5a: Fixed Feature Set

In this section, we constrain the movement of the arm so that the set of features tracked is invariant. This simplifies the tracking and allows us to concentrate on the baseline performance of the system. Note that this is the case assumed by many tracking systems [3], [27], [105].

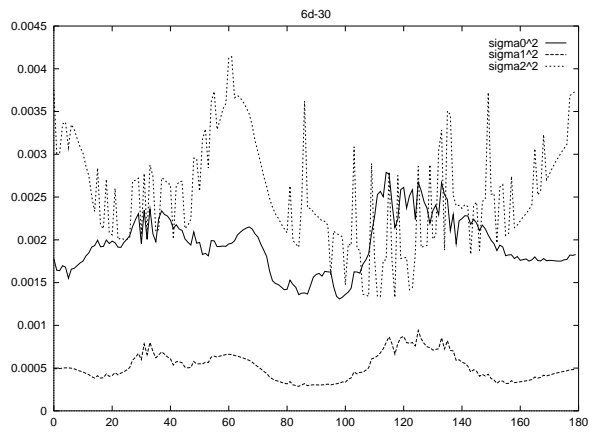
Figure 7.1(a) illustrates the evolution of the state estimate during tracking, with Figure 7.1(b) illustrating the uncertainty associated with this estimate. Figure 7.1(c) shows the expected visibility of each of the features throughout tracking.

#### 7.2 Case 5b: Self-Occlusion of Features

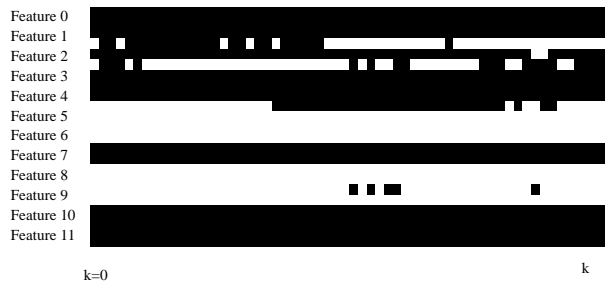
In this section, we illustrate how the system handles self-occlusion of features. An overview of the theory of this operation is given in Section 4.5.2. For example, in certain configurations



(a) State Estimates



(b) Uncertainty Estimates



(c) Feature Visibility Estimates

**Figure 7.1** Tracking results for Case 5a.

of the object, the upper arm link of the robot is between the camera and the gripper. In these configurations, any features on the gripper would be said to be self-occluded.

Figure 7.2(a) illustrates the evolution of the state estimate during tracking, with Figure 7.2(b) illustrating the uncertainty associated with this estimate. Figure 7.2(c) shows the expected visibility of each of the features throughout tracking.

### **7.3 Case 5c: External Occlusion of Features**

In this section, we describe the treatment of external occlusion, such as described in Section 4.5.3. Recall that external occlusion is defined as occlusion of features that cannot be predicted from object models, such as a person walking between the camera and the tracked object.

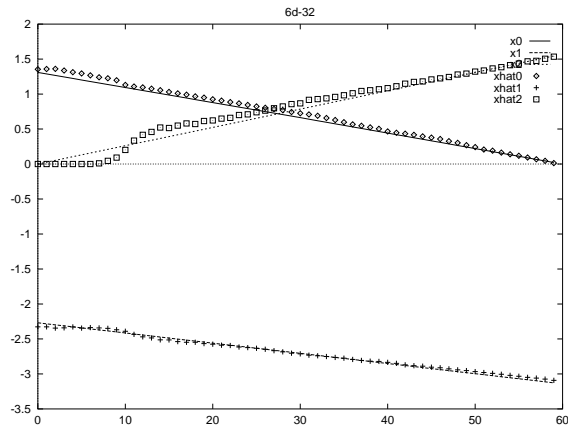
Figure 7.3(a) illustrates the evolution of the state estimate during tracking, with Figure 7.3(b) illustrating the uncertainty associated with this estimate. Figure 7.3(c) shows the expected visibility of each of the features throughout tracking.

### **7.4 Case 5d: Three DOF PUMA Arm**

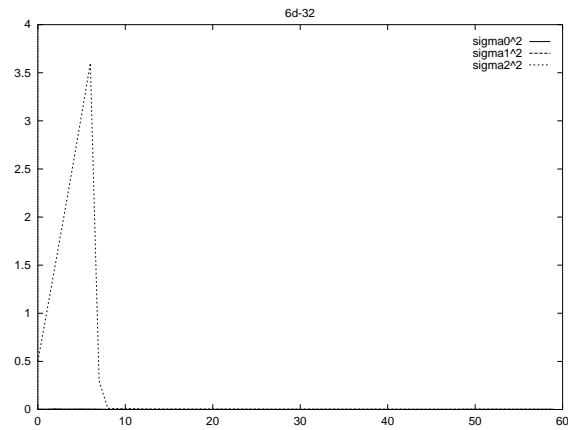
When the system is in full operation, the features described in the previous sections operate as appropriate, yielding an object-tracking system that can track features from widely varying viewpoints and can intelligently deal with the appearance and disappearance of features on an object. In this section, we illustrate this with a tracking example without the restrictions imposed above.

Figure 7.4(a) illustrates the evolution of the state estimate during tracking, with Figure 7.4(b) illustrating the uncertainty associated with this estimate. Figure 7.4(c) shows the expected visibility of each of the features throughout tracking.

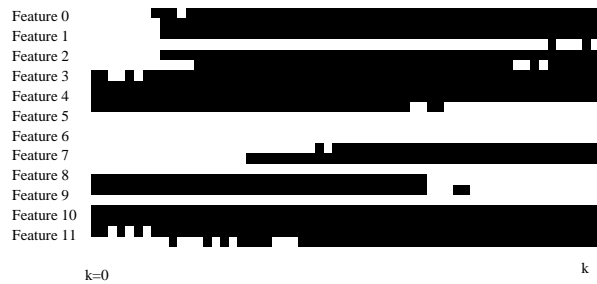




(a) State Estimates

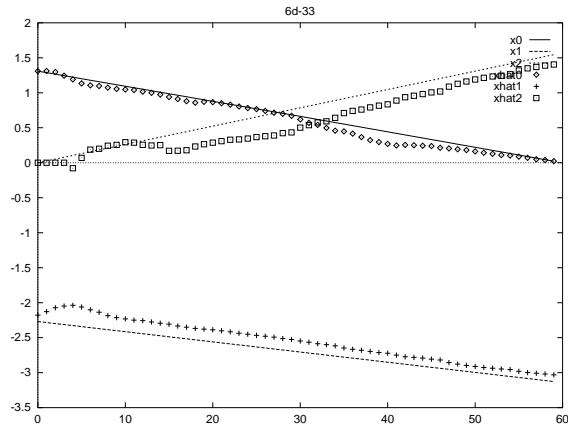


(b) Uncertainty Estimates

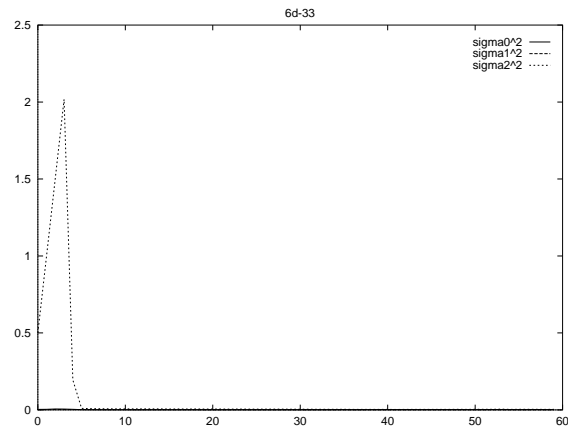


(c) Feature Visibility Estimates

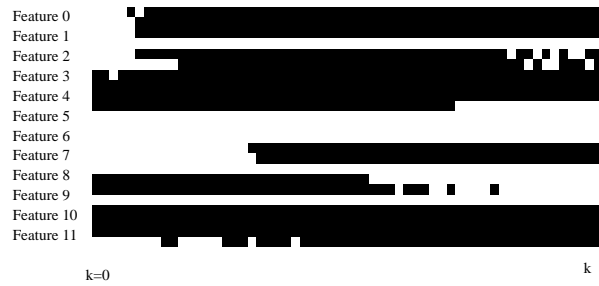
**Figure 7.2** Tracking results for Case 5b.



(a) State Estimates

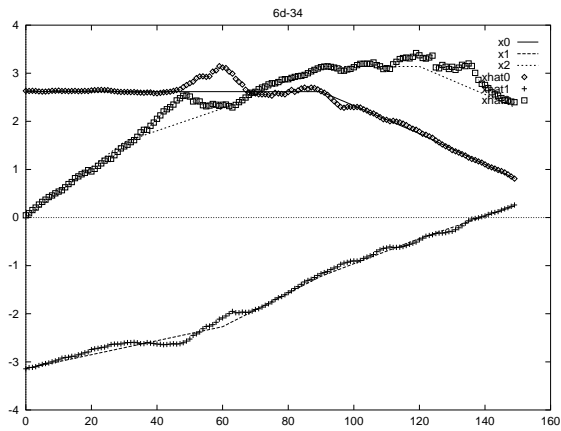


(b) Uncertainty Estimates

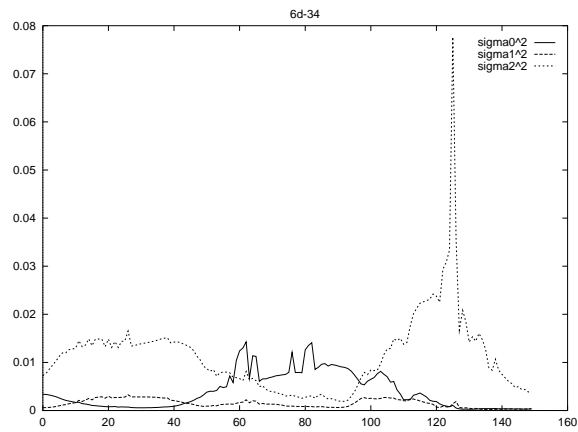


(c) Feature Visibility Estimates

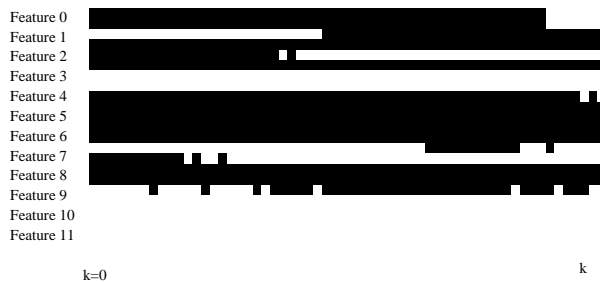
**Figure 7.3** Tracking results for Case 5c.



(a) State Estimates



(b) Uncertainty Estimates



(c) Feature Visibility Estimates

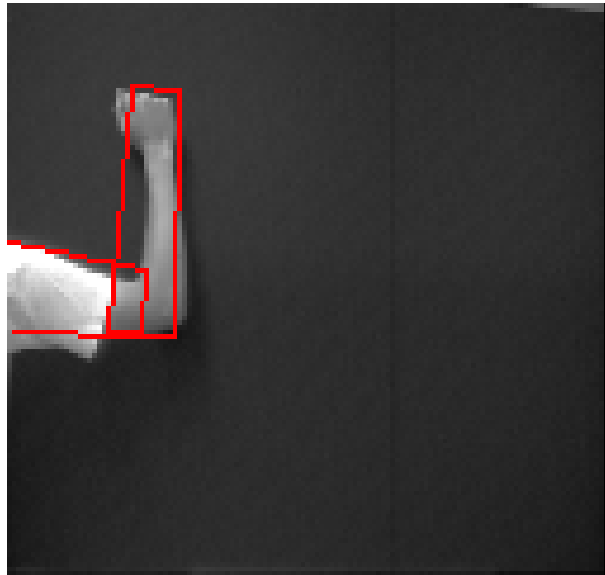
**Figure 7.4** Tracking results for Case 5d.

## 7.5 Case 6: Two DOF Arm With Unknown Link Lengths

In this case, the geometric model for the arm is not completely known. In particular, the lengths of the two links of the arm are unknown. The filter equations for this case, and the general theory behind the tracking, is described in Section 6.6. This section illustrates the system behavior in this case.

### 7.5.1 Case 6a: Correct link lengths

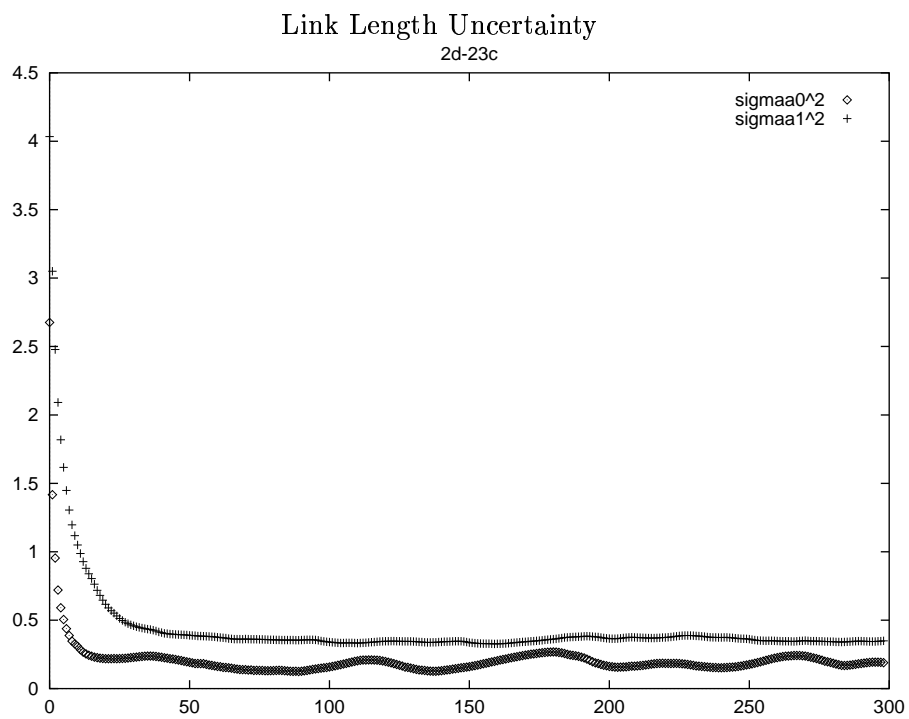
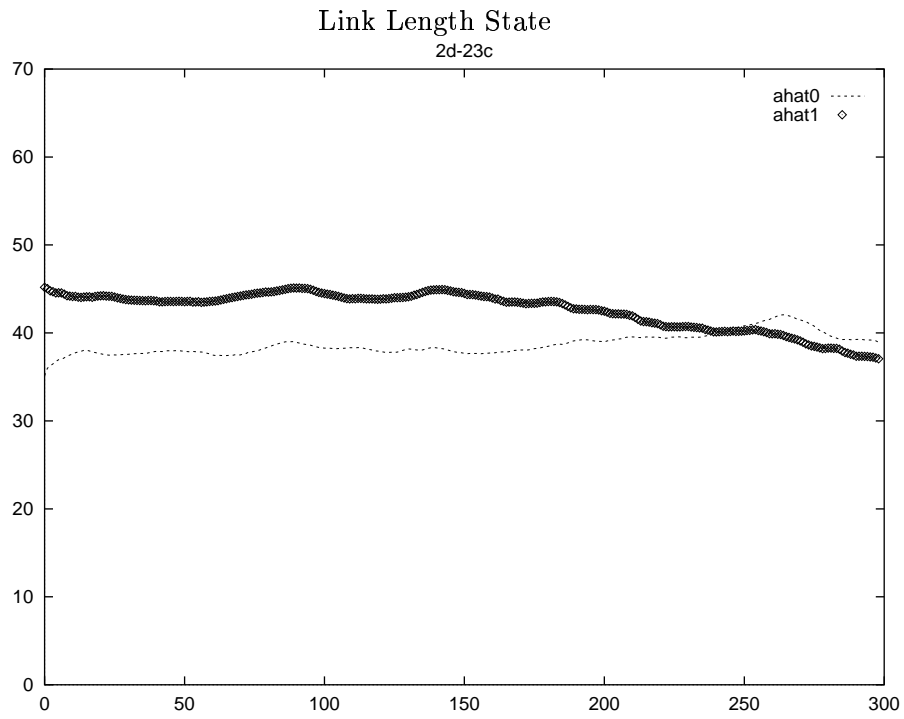
In this section, the lengths of the arm are initialized at approximately the correct values for the object given. Figure 7.5 illustrates the initial conditions of the filter with respect to the initial conditions of the actual arm, and Figures 7.6 and 7.7 show the tracking behavior of the system.



**Figure 7.5** Initial conditions for Case 6a.

### 7.5.2 Case 6b: Incorrect link lengths

In this experiment, the lengths of the arm and the initial joint angles are initialized in substantially incorrect values. Figure 7.8 illustrates the initial conditions of the filter with



**Figure 7.6** Tracking results for Case 6a: Link lengths.

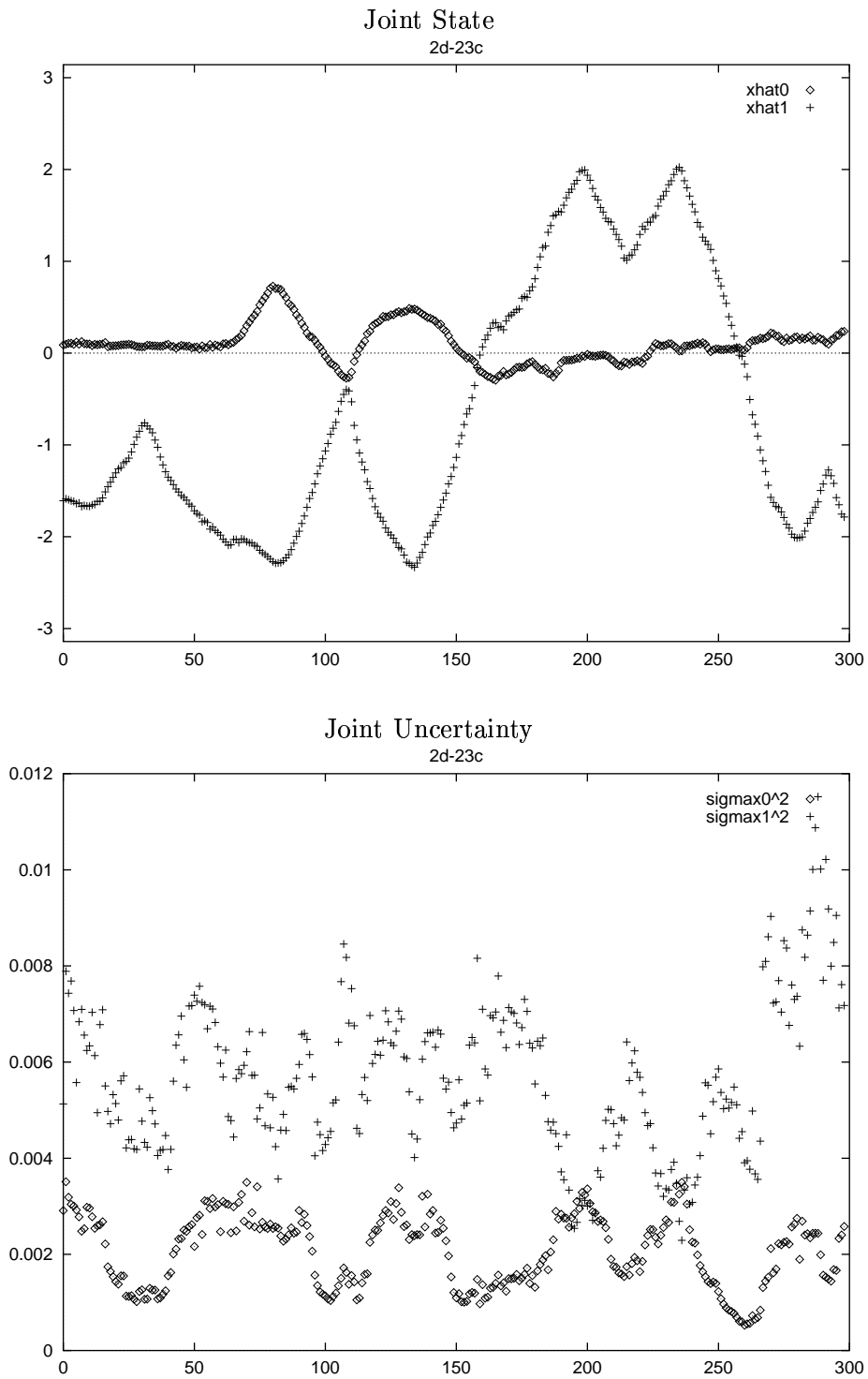
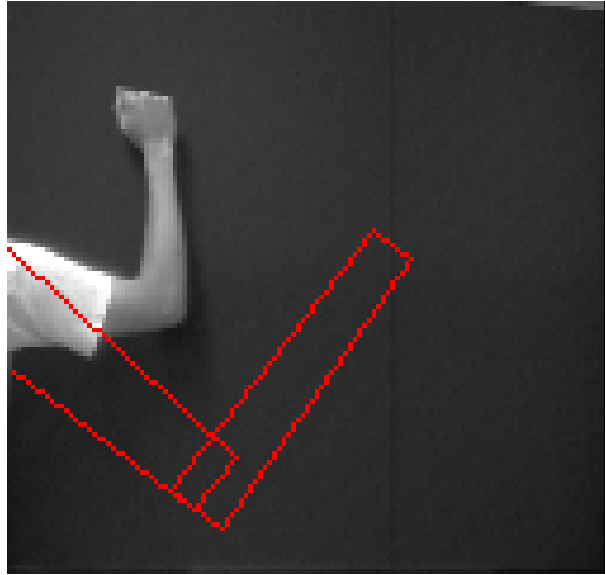


Figure 7.7 Tracking results for Case 6a: Joint angles.

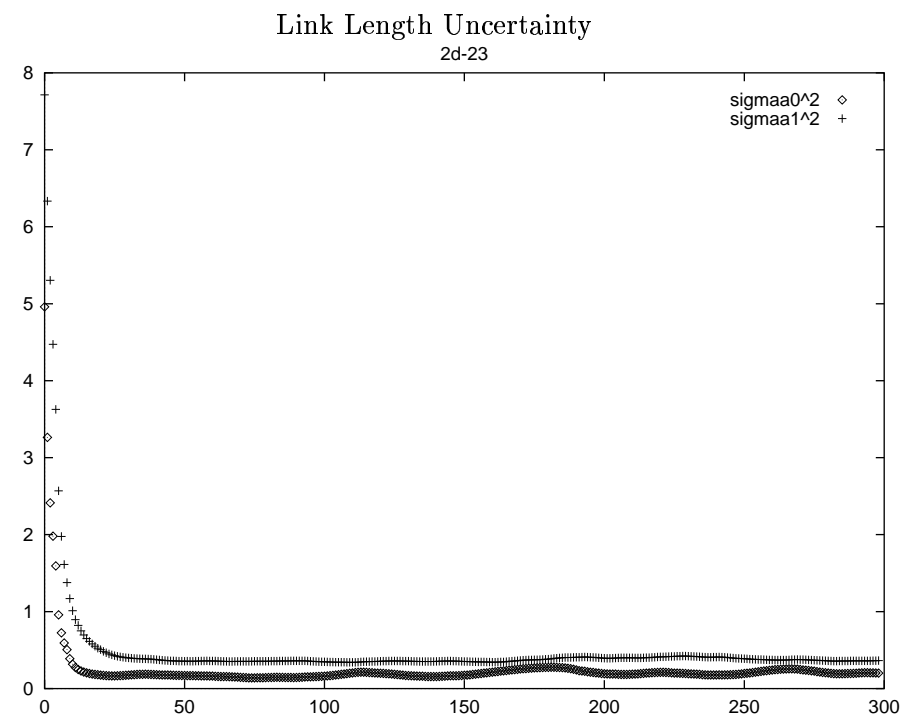
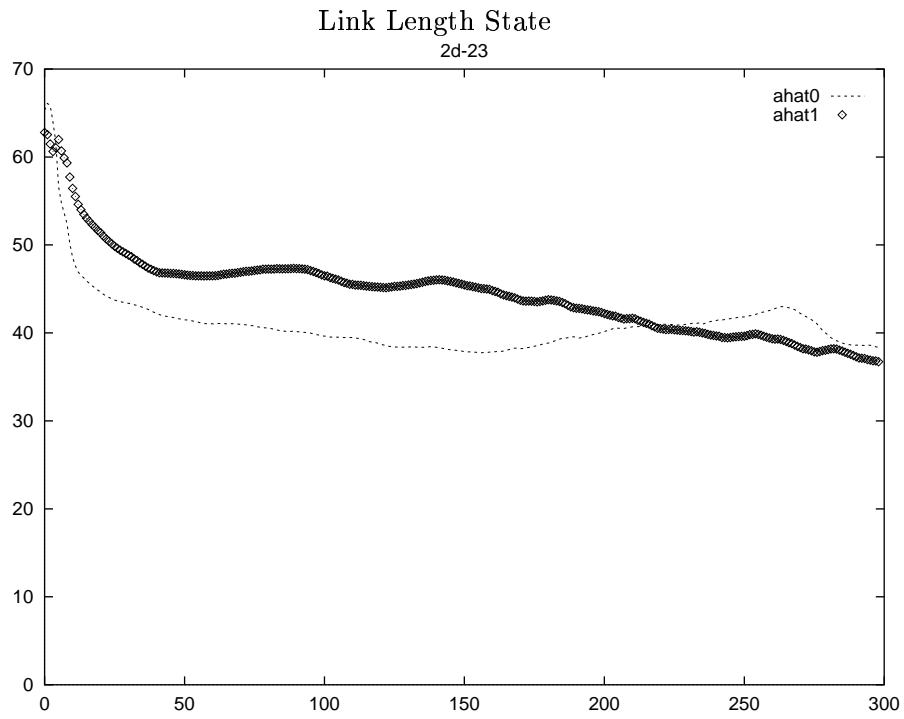
respect to the initial conditions of the actual arm, and Figures 7.9 and 7.10 show the tracking behavior of the system. In this experiment, the joint angles of the arm do not change for approximately the first 15 frames of the sequence.



**Figure 7.8** Initial conditions for Case 6b.

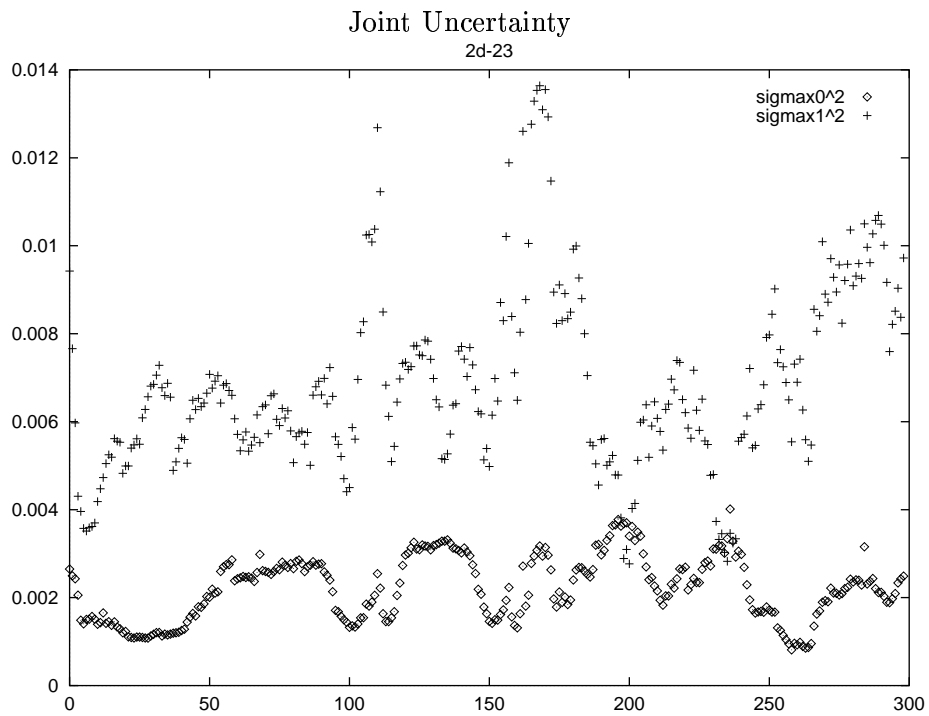
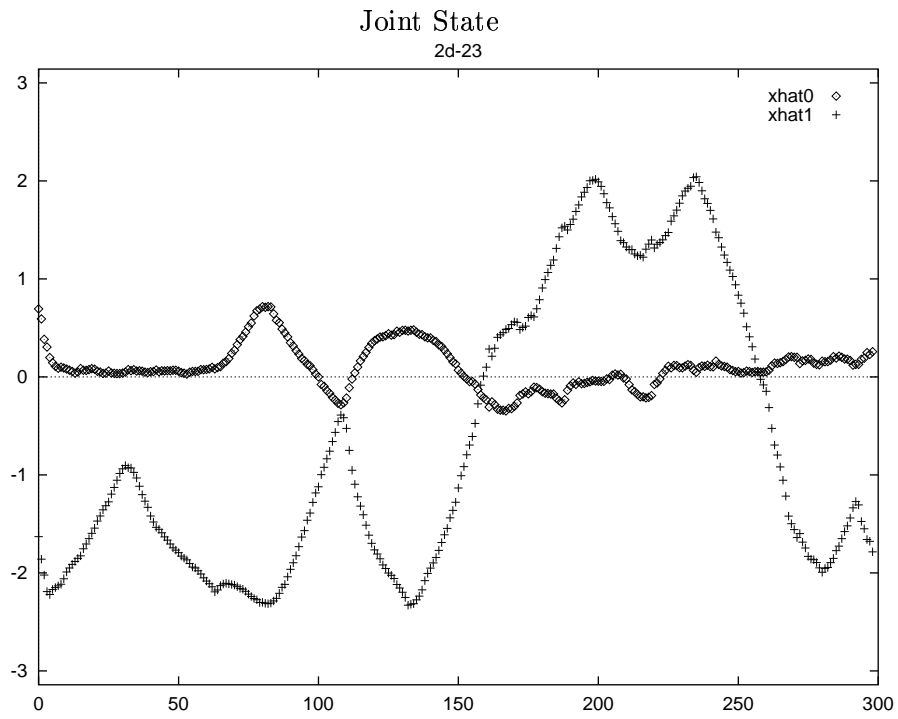
### **7.5.3 Case 6c: Incorrect link lengths, initial motion**

In this experiment, the lengths of the arm and the initial joint angles are initialized in substantially incorrect values. Figure 7.11 illustrates the initial conditions of the filter with respect to the initial conditions of the actual arm, and Figures 7.12 and 7.13 show the tracking behavior of the system. In this experiment, the joint angles of the arm are moving from the first frame.

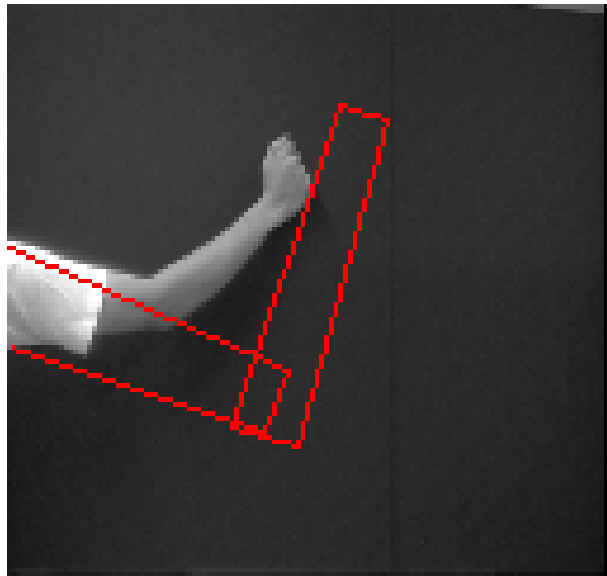


**Figure 7.9** Tracking results for Case 6b: Link lengths.

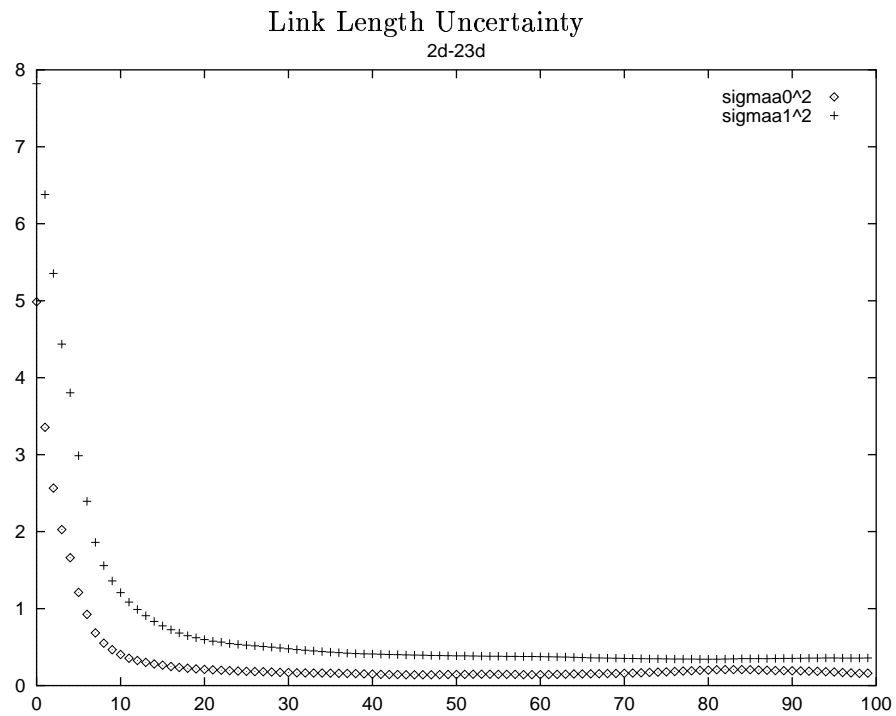
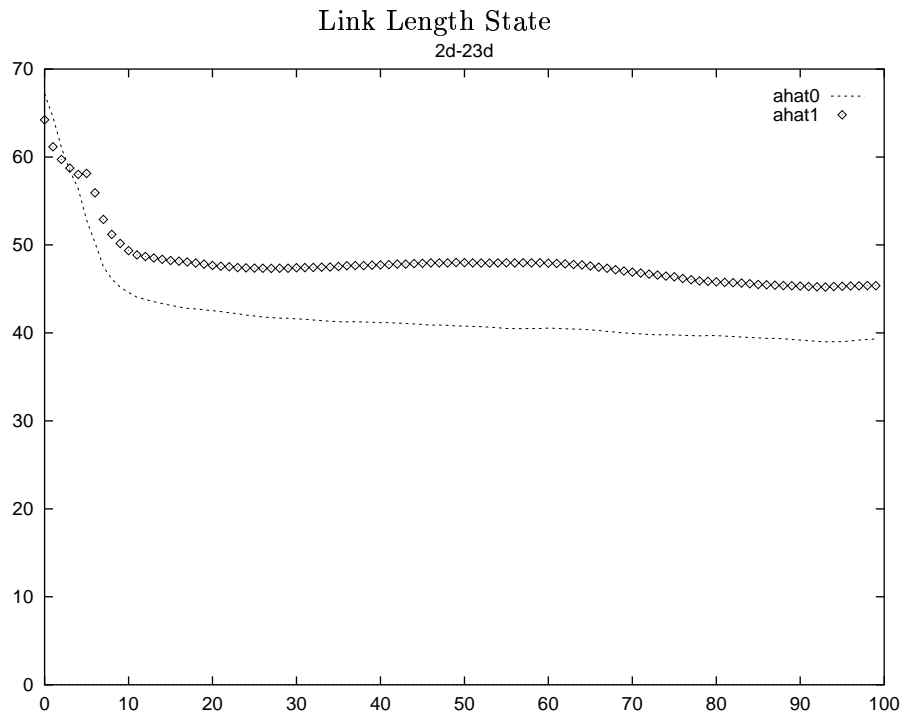




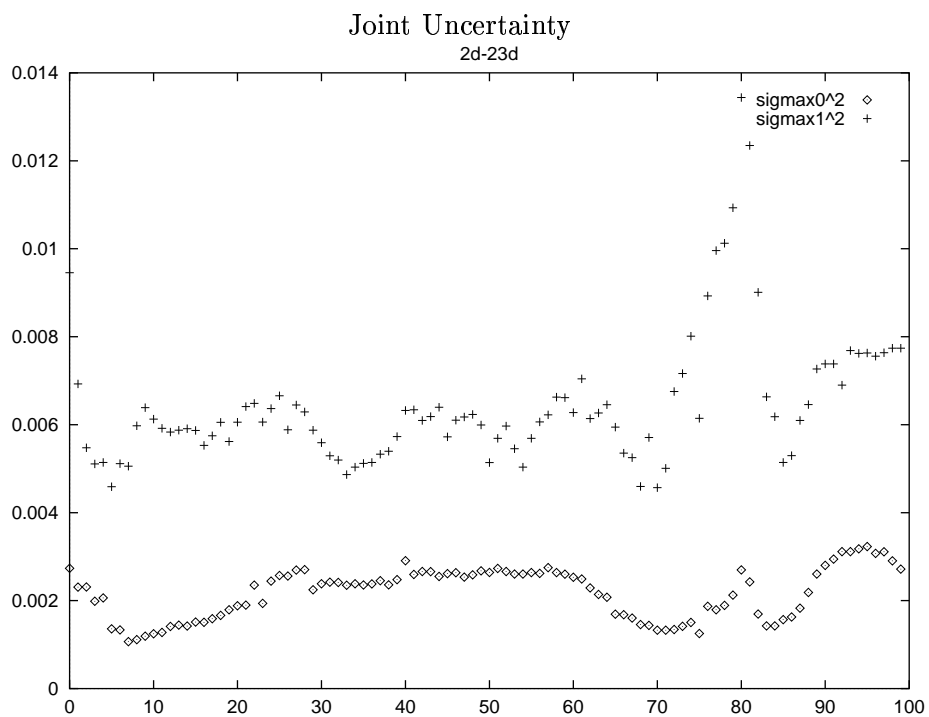
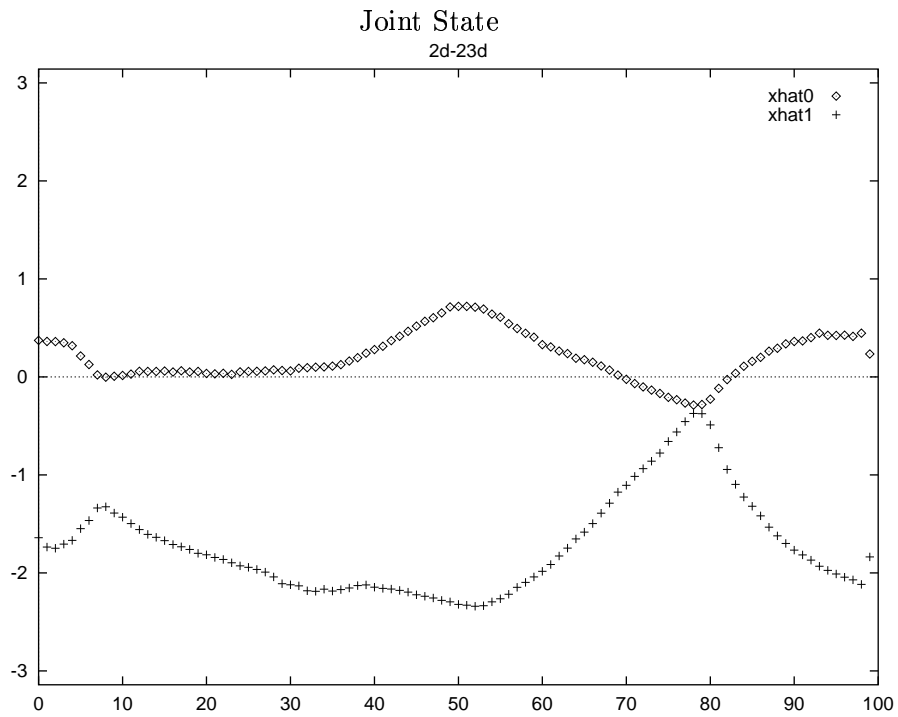
**Figure 7.10** Tracking results for Case 6b: Joint angles.



**Figure 7.11** Initial conditions for Case 6c.



**Figure 7.12** Tracking results for Case 6c: Link lengths.



**Figure 7.13** Tracking results for Case 6c: Joint angles.

## CHAPTER 8

### CONCLUSIONS AND FUTURE WORK

#### 8.1 Conclusions

This thesis presents a model-based object-tracking system capable of tracking complex articulated objects on the basis of monocular grayscale images of that object as it moves. The object model at the heart of the system is assumed to be known *a priori*, and is assumed to be perfect. In certain cases (see Section 6.6) this restriction can be relaxed by parameterizing unknown facets of the object model.

We have described the synergy that exists between the object model and the low-level feature tracking used in the system. The optimal filtering framework exploits the object model to overconstrain the feature trackers to operate in a manner consistent with the known kinematics of the object under consideration, so that the inaccurate feature-tracking results are not fatal to the system. The feature-tracking results are used in turn to optimally update the state vector, so that the object model is incrementally brought into accordance with observed data. This system combines the top-down approach of imposing an integral object model on the observed data with the bottom-up approach of using observed data to modify the assumptions of the system in a mathematically robust framework.

The use of spatial certainty measures allows the feature-tracking algorithm to convey the quality of the feature-tracking results to the object-tracking algorithm. Thus, the information that the feature-tracking algorithm obtains can be used to its maximum effectiveness. A sys-

tem that only extracts the location of the minimum SSD score will not have this information available, and will implicitly assign the same confidence to every feature tracking result.

Several features of this tracking system are novel. We have developed a method for generating feature templates for complex features from widely varying viewpoints [106], a method for estimating the reliability of feature tracking results during tracking [107], and a characterization of the use of the object models in considering what point feature motion reveals about object motion in images [108]. Each of these represents a contribution to the field of computer vision, and an advance in the state of the art in using object geometric models in tracking complex articulated objects.

## 8.2 Future Work

The framework introduced in this thesis is general and flexible, and could be applied in many situations not covered in this thesis. However, there are many portions of the system that have not been developed to their full extent and that could benefit from more sophisticated techniques. In this section, we will investigate some directions these extensions could take, as well as some novel situations that could benefit from such a general framework.

One aspect of the system that could be extended is the choice for the search area and template size. Currently, a fixed size rectangular search area and fixed size rectangular template are used, regardless of the size and expected motion of the feature. Knowledge about the object geometric model and expected feature motion could be used to define an image region most likely to contain the feature, for the purpose of using as a search region [39]. Similarly, knowledge about the relative size of a feature could be used to choose an appropriate template size.

The extension of the observation equations to the case where there are multiple cameras is straightforward. This would allow better 3D feature localization, and improved accuracy in tracking. This would be particularly helpful in cases like the PUMA described in Section 6.5 where the workspace is large relative to the feature size. The observation equations could also be parameterized to allow active vision. For example, the tracking results from the proximal

three joints of the PUMA could be used to track the position of the wrist of the PUMA. Then an active camera with a longer focal length could be trained on that spot, and the smaller links on the distal end of the PUMA could be tracked in this camera.

Some operations described in this thesis could benefit from hardware acceleration. There is hardware on the market today that could provide the graphics rendering described in Section 4.2.3 in real time. The SSD correlation process described in Section 4.2.4 is quite computationally expensive, but can be done in hardware in real time. In this thesis, all operations were performed sequentially on a Sparc 10 or Sparc 20 workstation, resulting in a processing rate of approximately 30 seconds per frame.

This framework is easily adaptable to the case where an optimal object trajectory is known. This would allow noncontact monitoring in certain situations of articulated object under external control. This has many potential applications in the fields of visual servoing, automated assembly, augmented reality, and human-computer interaction.

Kalman filtering as a framework for visual tracking has been shown to be a powerful tool in cases where object models are known. It optimally combines observations from disparate sources to update object models, making the system robust to feature mistracking and allowing the use of all information that is available from the feature tracking to be used. This synergy between object modeling and feature tracking results in increased performance compared to feature tracking alone.

## APPENDIX A

### CASE 1: BASE CASE

This appendix presents an annotated Maple session. This session derives the observation equation  $\mathbf{h}_k(\mathbf{x}_k)$  and dynamic model equation  $\mathbf{f}_k(\mathbf{x}_k)$  for the case presented.

Case 1

1 DOF Arm

Constant Position Dynamic Model

Orthogonal Projection

```
> with(linalg):
```

```
Warning, new definition for norm
```

```
Warning, new definition for trace
```

```
> a := vector(1); x := vector(1);
```

```
a := array(1..1, [])
```

```
x := array(1..1, [])
```

#### A.1 Projection Equations

s is the scale factor for the orthogonal projection

```
> s := 1: ortho := p -> vector([s*p[1]]);
```



$$ortho := p \rightarrow \text{vector}([s \ p_1])$$

Define observation function - mapping from joint angles to image plane coordinates.

```
> h := proc(x::vector) global a: ortho(vector([a[1]*sin(x)])): end;
```

```
h := proc(x::vector) global a; ortho(vector([a[1] * sin(x)])) end
```

## A.2 Partial Differentials

```
> H := diff(h(x)[1],x);
```

$$H := a_1 \cos(x)$$

## A.3 Dynamic Model

Define dynamic model

```
> f := x -> x;
```

$$f := x \rightarrow x$$

## A.4 Partial Differentials

```
> F := diff(f(x),x);
```

$$F := 1$$

## APPENDIX B

### CASE 2: ELEVATED CAMERA

This appendix presents an annotated Maple session. This session derives the observation equation  $\mathbf{h}_k(\mathbf{x}_k)$  and dynamic model equation  $\mathbf{f}_k(\mathbf{x}_k)$  for the case presented.

Case 1a - More Complex Models

One Link Arm

Const Position Dynamic Model

Perspective Projection - 2D sensor plane

#### B.1 Projection Equations

First, some setup.

```
> interface(labelling=false): a := 'a': c := 'c': x := 'x' : z := 'z': f
:= 'f' : d := 'd':
```

##### B.1.1 Transform derivation from end-effector coordinates to base coordinates

The standard homogeneous transformation matrix for the first link

```
> A[1] := matrix([[cos(q[1]),-sin(q[1]),0,a[1]*cos(q[1])],
[sin(q[1]),cos(q[1]),0,a[1]*sin(q[1])],
[0,0,1,0],
[0,0,0,1]]);
```

$$A_1 := \begin{bmatrix} \cos(q_1) & -\sin(q_1) & 0 & a_1 \cos(q_1) \\ \sin(q_1) & \cos(q_1) & 0 & a_1 \sin(q_1) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

### B.1.2 Derivation of transform from base coordinates to the camera frame

The full transformation is composed of three individual ones. This first one translates along the camera Z axis by  $x[c]$

```
> T_z := matrix([[1,0,0,0],[0,1,0,0],[0,0,1,sqrt(x[c]^2+z[c]^2)],[0,0,0,1]]);
```

$$T_z := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \sqrt{x_c^2 + z_c^2} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The angle between the workspace plane and the camera (between the world X axis and the camera Z axis)

```
> alpha := arctan(z[c],x[c]);
```

$$\alpha := \arctan(z_c, x_c)$$

After the translation, there is a rotation about the X axis by  $\alpha + \text{Pi}/2$  to make the Z axis line up with the world Z axis. Note that we rotate ccw about the X axis, to this is a negative number.

```
> R_alpha := matrix([[1,0,0,0],[0,cos(-(alpha+Pi/2)),-sin(-(alpha+Pi/2)),0],
% [0,sin(-(alpha+Pi/2)),cos(-(alpha+Pi/2)),0],[0,0,0,1]]);
```

$$R\_alpha := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -\frac{z_c}{\sqrt{\%1}} & \frac{x_c}{\sqrt{\%1}} & 0 \\ 0 & -\frac{x_c}{\sqrt{\%1}} & -\frac{z_c}{\sqrt{\%1}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\%1 := x_c^2 + z_c^2$$

Finally, we rotate about the Z axis by  $\text{Pi}/2$  to make the X axes coincide.

```
> R_z := matrix([[0,-1,0,0],[1,0,0,0],[0,0,1,0],[0,0,0,1]]);
```

$$R\_z := \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

To transform points from the world frame to the camera frame, left-multiply by this.

```
> T0c := evalm(T_z&*R_alpha&*R_z);
```

$$T0c := \begin{bmatrix} 0 & -1 & 0 & 0 \\ -\frac{z_c}{\sqrt{\%1}} & 0 & \frac{x_c}{\sqrt{\%1}} & 0 \\ -\frac{x_c}{\sqrt{\%1}} & 0 & -\frac{z_c}{\sqrt{\%1}} & \sqrt{\%1} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\%1 := x_c^2 + z_c^2$$

Just a sanity check.

```
> evalm(T0c&*vector([x,y,z,1]));
```

$$\left[ -y, -\frac{z_c x}{\sqrt{\%1}} + \frac{x_c z}{\sqrt{\%1}}, -\frac{x_c x}{\sqrt{\%1}} - \frac{z_c z}{\sqrt{\%1}} + \sqrt{\%1}, 1 \right]$$

$\%1 := x_c^2 + z_c^2$

### B.1.3 Combine and add perspective projection

This is the elbow location, in camera coordinates

```
> evalm(T0c&*A[1]&*vector([0,0,0,1]));
```

$$\left[ -a_1 \sin(q_1), -\frac{z_c a_1 \cos(q_1)}{\sqrt{x_c^2 + z_c^2}}, -\frac{x_c a_1 \cos(q_1)}{\sqrt{x_c^2 + z_c^2}} + \sqrt{x_c^2 + z_c^2}, 1 \right]$$

Full elbow mapping (into camera coordinates), as procedure

```
> h_elbow := (q) -> vector([-a[1]*sin(q[1]),
-z[c]/(x[c]^2+z[c]^2)^(1/2)*a[1]*cos(q[1]),
-x[c]/(x[c]^2+z[c]^2)^(1/2)*a[1]*cos(q[1])+(x[c]^2+z[c]^2)^(1/2),
1]);
```

$$h\_elbow := q \rightarrow \text{vector} \left( \left[ -a_1 \sin(q_1), -\frac{z_c a_1 \cos(q_1)}{\sqrt{x_c^2 + z_c^2}}, -\frac{x_c a_1 \cos(q_1)}{\sqrt{x_c^2 + z_c^2}} + \sqrt{x_c^2 + z_c^2}, 1 \right] \right)$$

Given a vector [x,y,z,1] in camera coordinates compute the projection onto the image plane

(Standard projection equations)

```
> perspective := proc(p::vector) vector([-1/f*p[1]/p[3], -1/f*p[2]/p[3]]): end:
ortho := proc(p::vector) vector([-1/f*p[1], -1/f*p[2]]): end:
```

This is the full mapping from configuration space to image plane space

```
> #h := (q1) -> vector([perspective(h_elbow(vector([q1])))[1],
# perspective(h_elbow(vector([q1])))[2]]);
h := (q1) -> vector([ortho(h_elbow(vector([q1])))[1],
ortho(h_elbow(vector([q1])))[2]]);
```

$$h := q1 \rightarrow \text{vector}([\text{ortho}(\text{h\_elbow}(\text{vector}([q1])))_1, \text{ortho}(\text{h\_elbow}(\text{vector}([q1])))_2])$$

Let's see what this looks like symbolically

```
> h(q[1]);
```

$$\left[ \frac{a_1 \sin(q_1)}{f}, \frac{z_c a_1 \cos(q_1)}{f \sqrt{x_c^2 + z_c^2}} \right]$$

## B.2 Partial Differentials

The partials of this function are as follows.

```
> diff(h(q[1])[1],q[1]); dh1dq1 := unapply(",q[1]):
```

$$\frac{a_1 \cos(q_1)}{f}$$

```
> diff(h(q[1])[2],q[1]); dh2dq1 := unapply(",q[1]):
```

$$-\frac{z_c a_1 \sin(q_1)}{f \sqrt{x_c^2 + z_c^2}}$$

## B.3 Dynamic Model

$f$  is defined as the estimate for the state, given the (estimated) current state. We use a constant position model, with sampling interval  $D$

```
> f := (xhat) -> xhat;
```

$$f := xhat \rightarrow xhat$$

## B.4 Partial Differentials

```
> diff(f(q)[1],q[1]); df1dq1 := unapply(",q[1],q[2]):
```

## APPENDIX C

### CASE 3: MORE COMPLEX MODELS

This appendix presents an annotated Maple session. This session derives the observation equation  $\mathbf{h}_k(\mathbf{x}_k)$  and dynamic model equation  $\mathbf{f}_k(\mathbf{x}_k)$  for the case presented.

Case 2 - More Complex Models

Two Link Arm

Const Position Dynamic Model

Perspective Projection

#### C.1 Projection Equations

First, some setup.

```
> interface(labelling=false): a := 'a': c := 'c': x := 'x' : z := 'z': f
:= 'f' : d := 'd':
```

##### C.1.1 Transform derivation from end-effector coordinates to base coordinates

The standard homogeneous transformation matrix for the first link

```
> A[1] :=
matrix([[cos(q[1]), -sin(q[1]), 0, a[1]*cos(q[1])],
[sin(q[1]), cos(q[1]), 0, a[1]*sin(q[1])],
```

```
[0,0,1,0],
[0,0,0,1]]);
```

$$A_1 := \begin{bmatrix} \cos(q_1) & -\sin(q_1) & 0 & a_1 \cos(q_1) \\ \sin(q_1) & \cos(q_1) & 0 & a_1 \sin(q_1) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The standard homogeneous transformation matrix for the second link

```
> A[2] :=
matrix([[cos(q[2]), -sin(q[2]), 0, a[2]*cos(q[2])],
[ sin(q[2]), cos(q[2]), 0, a[2]*sin(q[2])],
[0,0,1,0],
[0,0,0,1]]);
```

$$A_2 := \begin{bmatrix} \cos(q_2) & -\sin(q_2) & 0 & a_2 \cos(q_2) \\ \sin(q_2) & \cos(q_2) & 0 & a_2 \sin(q_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Combine them.

```
> A[12] := evalm(A[1]*A[2]);
```



$$\begin{aligned}
A_{12} := & \\
& \left[ \cos(q_1) \cos(q_2) - \sin(q_1) \sin(q_2), -\cos(q_1) \sin(q_2) - \sin(q_1) \cos(q_2), 0, \right. \\
& \left. \cos(q_1) a_2 \cos(q_2) - \sin(q_1) a_2 \sin(q_2) + a_1 \cos(q_1) \right] \\
& \left[ \sin(q_1) \cos(q_2) + \cos(q_1) \sin(q_2), \cos(q_1) \cos(q_2) - \sin(q_1) \sin(q_2), 0, \right. \\
& \left. \sin(q_1) a_2 \cos(q_2) + \cos(q_1) a_2 \sin(q_2) + a_1 \sin(q_1) \right] \\
& [0, 0, 1, 0] \\
& [0, 0, 0, 1]
\end{aligned}$$

Composite transformation from coordinate frame of end-effector to base

```
> T20 := map(combine,A[12],trig);
```

$$T_{20} := \begin{bmatrix} \cos(q_1 + q_2) & -\sin(q_1 + q_2) & 0 & a_2 \cos(q_1 + q_2) + a_1 \cos(q_1) \\ \sin(q_1 + q_2) & \cos(q_1 + q_2) & 0 & a_2 \sin(q_1 + q_2) + a_1 \sin(q_1) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

### C.1.2 Derivation of transform from base coordinates to the camera frame

The full transformation is composed of three individual ones. This first one translates along the camera Z axis by  $x[c]$

```
> T_z := matrix([[1,0,0,0],[0,1,0,0],[0,0,1,sqrt(x[c]^2+z[c]^2)],[0,0,0,1]]);
```

$$T_z := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \sqrt{x_c^2 + z_c^2} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The angle between the workspace plane and the camera (between the world X axis and the camera Z axis)

```
> alpha := arctan(z[c],x[c]);
```

$$\alpha := \arctan(z_c, x_c)$$

After the translation, there is a rotation about the X axis by  $\alpha + \pi/2$  to make the Z axis line up with the world Z axis. Note that we rotate ccw about the X axis, so this is a negative number.

```
> R_alpha :=
matrix([[1,0,0,0],[0,cos(-(alpha+Pi/2)),-sin(-(alpha+Pi/2)),0],
[0,sin(-(alpha+Pi/2)),cos(-(alpha+Pi/2)),0],
[0,0,0,1]]);
```

$$R\_alpha := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -\frac{z_c}{\sqrt{\%1}} & \frac{x_c}{\sqrt{\%1}} & 0 \\ 0 & -\frac{x_c}{\sqrt{\%1}} & -\frac{z_c}{\sqrt{\%1}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\%1 := x_c^2 + z_c^2$$

Finally, we rotate about the Z axis by  $\pi/2$  to make the X axes coincide.

```
> R_z := matrix([[0,-1,0,0],[1,0,0,0],[0,0,1,0],[0,0,0,1]]);
```

$$R_z := \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

To transform points from the world frame to the camera frame, left-multiply by this.

```
> T0c := evalm(T_z&*R_alpha&*R_z);
```

$$T0c := \begin{bmatrix} 0 & -1 & 0 & 0 \\ -\frac{z_c}{\sqrt{\%1}} & 0 & \frac{x_c}{\sqrt{\%1}} & 0 \\ -\frac{x_c}{\sqrt{\%1}} & 0 & -\frac{z_c}{\sqrt{\%1}} & \sqrt{\%1} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$\%1 := x_c^2 + z_c^2$

Just a sanity check.

```
> evalm(T0c&*vector([x,y,z,1]));
```

$$\left[ -y, -\frac{z_c x}{\sqrt{\%1}} + \frac{x_c z}{\sqrt{\%1}}, -\frac{x_c x}{\sqrt{\%1}} - \frac{z_c z}{\sqrt{\%1}} + \sqrt{\%1}, 1 \right]$$

$\%1 := x_c^2 + z_c^2$

### C.1.3 Combine and add perspective projection

This is the end-effector location, in camera coordinates

```
> evalm(T0c&*T20&*vector([0,0,0,1]));
```

$$\left[ \begin{array}{l} -a_2 \sin(q_1 + q_2) - a_1 \sin(q_1), -\frac{z_c (a_2 \cos(q_1 + q_2) + a_1 \cos(q_1))}{\sqrt{x_c^2 + z_c^2}}, \\ -\frac{x_c (a_2 \cos(q_1 + q_2) + a_1 \cos(q_1))}{\sqrt{x_c^2 + z_c^2}} + \sqrt{x_c^2 + z_c^2}, 1 \end{array} \right]$$

This is the elbow location, in camera coordinates

```
> evalm(T0c&*A[1]&*vector([0,0,0,1]));
```

$$\left[ -a_1 \sin(q_1), -\frac{z_c a_1 \cos(q_1)}{\sqrt{x_c^2 + z_c^2}}, -\frac{x_c a_1 \cos(q_1)}{\sqrt{x_c^2 + z_c^2}} + \sqrt{x_c^2 + z_c^2}, 1 \right]$$

Full end effector mapping (into camera coordinates), as procedure

```
> h_endeffector := (q) ->
vector([-a[2]*sin(q[1]+q[2])-a[1]*sin(q[1]),
-z[c]/(x[c]^2+z[c]^2)^(1/2)*(a[2]*cos(q[1]+q[2])+a[1]*cos(q[1])),
-x[c]/(x[c]^2+z[c]^2)^(1/2)*(a[2]*cos(q[1]+q[2])+a[1]*cos(q[1]))
+(x[c]^2+z[c]^2)^(1/2),
1]);
```

$$h\_endeffector := q \rightarrow \text{vector} \left( \left[ \begin{array}{l} -a_2 \sin(q_1 + q_2) - a_1 \sin(q_1), -\frac{z_c (a_2 \cos(q_1 + q_2) + a_1 \cos(q_1))}{\sqrt{x_c^2 + z_c^2}}, \\ -\frac{x_c (a_2 \cos(q_1 + q_2) + a_1 \cos(q_1))}{\sqrt{x_c^2 + z_c^2}} + \sqrt{x_c^2 + z_c^2}, 1 \end{array} \right] \right)$$

Full elbow mapping (into camera coordinates), as procedure

```
> h_elbow := (q) -> vector([-a[1]*sin(q[1]),
-z[c]/(x[c]^2+z[c]^2)^(1/2)*a[1]*cos(q[1]),
-x[c]/(x[c]^2+z[c]^2)^(1/2)*a[1]*cos(q[1])+(x[c]^2+z[c]^2)^(1/2),
1]);
```

$$h\_elbow := q \rightarrow \text{vector} \left( \left[ \begin{array}{l} -a_1 \sin(q_1), -\frac{z_c a_1 \cos(q_1)}{\sqrt{x_c^2 + z_c^2}}, -\frac{x_c a_1 \cos(q_1)}{\sqrt{x_c^2 + z_c^2}} + \sqrt{x_c^2 + z_c^2}, 1 \end{array} \right] \right)$$

Given a vector [x,y,z,1] in camera coordinates compute the projection onto the image plane

(Standard projection equations)

```
> perspective := proc(p:vector) vector([-1/f*p[1]/p[3],-1/f*p[2]/p[3]]): end;
```

This is the full mapping from configuration space to image plane space

```

> h := (q1,q2) -> vector([perspective(h_elbow(vector([q1,q2])))[1],
perspective(h_elbow(vector([q1,q2])))[2],
perspective(h_endeffector(vector([q1,q2])))[1],
perspective(h_endeffector(vector([q1,q2])))[2]]);

```

```

h := (q1, q2) -> vector([perspective(h_elbow(vector([q1, q2])))_1,
perspective(h_elbow(vector([q1, q2])))_2, perspective(h_endeffector(vector([q1, q2])))_1,
perspective(h_endeffector(vector([q1, q2])))_2])

```

Let's see what this looks like symbolically

```

> h(q[1],q[2]);

```

$$\left[ \begin{array}{c} \frac{a_1 \sin(q_1)}{f \left( -\frac{x_c a_1 \cos(q_1)}{\sqrt{\%1}} + \sqrt{\%1} \right)}, \frac{z_c a_1 \cos(q_1)}{f \sqrt{\%1} \left( -\frac{x_c a_1 \cos(q_1)}{\sqrt{\%1}} + \sqrt{\%1} \right)}, -\frac{-a_2 \sin(q_1 + q_2) - a_1 \sin(q_1)}{f \left( -\frac{x_c \%2}{\sqrt{\%1}} + \sqrt{\%1} \right)}, \\ \frac{z_c \%2}{f \sqrt{\%1} \left( -\frac{x_c \%2}{\sqrt{\%1}} + \sqrt{\%1} \right)} \end{array} \right]$$

$\%1 := x_c^2 + z_c^2$   
 $\%2 := a_2 \cos(q_1 + q_2) + a_1 \cos(q_1)$

## C.2 Partial Differentials

The partials of this function are as follows.

```

> diff(h(q[1],q[2])[1],q[1]); dh1dq1 := unapply(",q[1],q[2]):

```

$$\frac{a_1 \cos(q_1)}{f \left( -\frac{x_c a_1 \cos(q_1)}{\sqrt{\%1}} + \sqrt{\%1} \right)} - \frac{a_1^2 \sin(q_1)^2 x_c}{f \left( -\frac{x_c a_1 \cos(q_1)}{\sqrt{\%1}} + \sqrt{\%1} \right)^2 \sqrt{\%1}}$$

$\%1 := x_c^2 + z_c^2$

```

> diff(h(q[1],q[2])[1],q[2]); dh1dq2 := unapply(",q[1],q[2]):

```

> diff(h(q[1],q[2])[2],q[1]); dh2dq1 := unapply(",q[1],q[2]):

$$-\frac{z_c a_1 \sin(q_1)}{f \sqrt{\%1} \left(-\frac{x_c a_1 \cos(q_1)}{\sqrt{\%1}} + \sqrt{\%1}\right)} - \frac{z_c a_1^2 \cos(q_1) x_c \sin(q_1)}{f \%1 \left(-\frac{x_c a_1 \cos(q_1)}{\sqrt{\%1}} + \sqrt{\%1}\right)^2}$$

$$\%1 := x_c^2 + z_c^2$$

> diff(h(q[1],q[2])[2],q[2]); dh2dq2 := unapply(",q[1],q[2]):

0

> diff(h(q[1],q[2])[3],q[1]); dh3dq1 := unapply(",q[1],q[2]):

$$-\frac{-a_2 \cos(q_1 + q_2) - a_1 \cos(q_1)}{f \left(-\frac{x_c (a_2 \cos(q_1 + q_2) + a_1 \cos(q_1))}{\sqrt{\%1}} + \sqrt{\%1}\right)}$$

$$-\frac{(-a_2 \sin(q_1 + q_2) - a_1 \sin(q_1))^2 x_c}{f \left(-\frac{x_c (a_2 \cos(q_1 + q_2) + a_1 \cos(q_1))}{\sqrt{\%1}} + \sqrt{\%1}\right)^2 \sqrt{\%1}}$$

$$\%1 := x_c^2 + z_c^2$$

> diff(h(q[1],q[2])[3],q[2]); dh3dq2 := unapply(",q[1],q[2]):

$$\frac{a_2 \cos(q_1 + q_2)}{f \left(-\frac{x_c (a_2 \cos(q_1 + q_2) + a_1 \cos(q_1))}{\sqrt{\%1}} + \sqrt{\%1}\right)}$$

$$+\frac{(-a_2 \sin(q_1 + q_2) - a_1 \sin(q_1)) x_c a_2 \sin(q_1 + q_2)}{f \left(-\frac{x_c (a_2 \cos(q_1 + q_2) + a_1 \cos(q_1))}{\sqrt{\%1}} + \sqrt{\%1}\right)^2 \sqrt{\%1}}$$

$$\%1 := x_c^2 + z_c^2$$

> diff(h(q[1],q[2])[4],q[1]); dh4dq1 := unapply(",q[1],q[2]):

$$\frac{z_c (-a_2 \sin(q_1 + q_2) - a_1 \sin(q_1))}{f \sqrt{\%1} \left(-\frac{x_c \%2}{\sqrt{\%1}} + \sqrt{\%1}\right)} + \frac{z_c \%2 x_c (-a_2 \sin(q_1 + q_2) - a_1 \sin(q_1))}{f \%1 \left(-\frac{x_c \%2}{\sqrt{\%1}} + \sqrt{\%1}\right)^2}$$

$$\%1 := x_c^2 + z_c^2$$

$$\%2 := a_2 \cos(q_1 + q_2) + a_1 \cos(q_1)$$

> diff(h(q[1],q[2])[4],q[2]); dh4dq2 := unapply(",q[1],q[2]):

$$-\frac{z_c a_2 \sin(q_1 + q_2)}{f \sqrt{\%1} \left(-\frac{x_c \%2}{\sqrt{\%1}} + \sqrt{\%1}\right)} - \frac{z_c \%2 x_c a_2 \sin(q_1 + q_2)}{f \%1 \left(-\frac{x_c \%2}{\sqrt{\%1}} + \sqrt{\%1}\right)^2}$$

$$\%1 := x_c^2 + z_c^2$$

$$\%2 := a_2 \cos(q_1 + q_2) + a_1 \cos(q_1)$$

### C.3 Dynamic Model

$f$  is defined as the estimate for the state, given the (estimated) current state. We use a constant position model, with sampling interval  $D$

```
> f := (xhat) -> xhat;
```

$$f := xhat \rightarrow xhat$$

### C.4 Partial Differentials

```
> diff(f(q)[1],q[1]); df1dq1 := unapply(",q[1],q[2]):
```

1

```
> diff(f(q)[1],q[2]); df1dq2 := unapply(",q[1],q[2]):
```

0

```
> diff(f(q)[2],q[1]); df2dq1 := unapply(",q[1],q[2]):
```

0

```
> diff(f(q)[2],q[2]); df2dq2 := unapply(",q[1],q[2]):
```

1

## APPENDIX D

### CASE 4: DYNAMIC MODEL

This appendix presents an annotated Maple session. This session derives the observation equation  $\mathbf{h}_k(\mathbf{x}_k)$  and dynamic model equation  $\mathbf{f}_k(\mathbf{x}_k)$  for the case presented.

#### 1 DOF Arm - 2D Sensor - Position and Velocity Estimates

We have seen how the EKF can be used as a mechanism for weighting the observations returned from a tracking system, using information about the configuration of the robot and imaging system to evaluate the quality of the observations when updating the estimated state of the system. In this example, we illustrate the use of a dynamic system model. We have a 1 link arm, and an elevated 2D sensor. We will include both  $q$  and  $dq/dt$  in our state vector, and use the velocity estimates in computing the predicted observation.

### D.1 Projection Equations

First, some setup.

```
> interface(labelling=false): a := 'a': c := 'c': x := 'x' : z := 'z': f
:= 'f' : d := 'd':
```

#### D.1.1 Transform derivation from end-effector coordinates to base coordinates

The standard homogeneous transformation matrix for the first link



```

> A[1] :=
matrix([[cos(q[1]),-sin(q[1]),0,a[1]*cos(q[1])],
[sin(q[1]),cos(q[1]),0,a[1]*sin(q[1])],
[0,0,1,0],
[0,0,0,1]]);

```

$$A_1 := \begin{bmatrix} \cos(q_1) & -\sin(q_1) & 0 & a_1 \cos(q_1) \\ \sin(q_1) & \cos(q_1) & 0 & a_1 \sin(q_1) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Composite transformation from coordinate frame of end-effector to base

```

> T20 := A[1];

```

$$T_{20} := A_1$$

### D.1.2 Derivation of transform from base coordinates to the camera frame

The full transformation is composed of three individual ones. This first one translates along the camera Z axis by  $x[c]$

```

> T_z := matrix([[1,0,0,0],[0,1,0,0],[0,0,1,sqrt(x[c]^2+z[c]^2)],[0,0,0,1]]);

```

$$T_z := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \sqrt{x_c^2 + z_c^2} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The angle between the workspace plane and the camera (between the world X axis and the camera Z axis)

```
> alpha := arctan(z[c],x[c]);
```

$$\alpha := \arctan(z_c, x_c)$$

After the translation, there is a rotation about the X axis by  $\alpha + \pi/2$  to make the Z axis line up with the world Z axis. Note that we rotate ccw about the X axis, so this is a negative number.

```
> R_alpha :=
matrix([[1,0,0,0],[0,cos(-(alpha+Pi/2)),-sin(-(alpha+Pi/2)),0],
[0,sin(-(alpha+Pi/2)),cos(-(alpha+Pi/2)),0],
[0,0,0,1]]);
```

$$R\_alpha := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -\frac{z_c}{\sqrt{\%1}} & \frac{x_c}{\sqrt{\%1}} & 0 \\ 0 & -\frac{x_c}{\sqrt{\%1}} & -\frac{z_c}{\sqrt{\%1}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\%1 := x_c^2 + z_c^2$$

Finally, we rotate about the Z axis by  $\pi/2$  to make the X axes coincide.

```
> R_z := matrix([[0,-1,0,0],[1,0,0,0],[0,0,1,0],[0,0,0,1]]);
```

$$R_z := \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

To transform points from the world frame to the camera frame, left-multiply by this.

```
> T0c := evalm(T_z&*R_alpha&*R_z);
```

$$T0c := \begin{bmatrix} 0 & -1 & 0 & 0 \\ -\frac{z_c}{\sqrt{\%1}} & 0 & \frac{x_c}{\sqrt{\%1}} & 0 \\ -\frac{x_c}{\sqrt{\%1}} & 0 & -\frac{z_c}{\sqrt{\%1}} & \sqrt{\%1} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$\%1 := x_c^2 + z_c^2$

Just a sanity check.

```
> evalm(T0c&*vector([x,y,z,1]));
```

$$\left[ -y, -\frac{z_c x}{\sqrt{\%1}} + \frac{x_c z}{\sqrt{\%1}}, -\frac{x_c x}{\sqrt{\%1}} - \frac{z_c z}{\sqrt{\%1}} + \sqrt{\%1}, 1 \right]$$

$\%1 := x_c^2 + z_c^2$

### D.1.3 Combine and add perspective projection

This is the end-effector location in camera coordinates

```
> evalm(T0c&*T20&*vector([0,0,0,1]));
```

$$\left[ -a_1 \sin(q_1), -\frac{z_c a_1 \cos(q_1)}{\sqrt{x_c^2 + z_c^2}}, -\frac{x_c a_1 \cos(q_1)}{\sqrt{x_c^2 + z_c^2}} + \sqrt{x_c^2 + z_c^2}, 1 \right]$$

End effector mapping (into camera coordinates) as procedure

```
> h_endeffector := (q) ->
vector([-a[1]*sin(q[1]), -z[c]/(x[c]^2+z[c]^2)^(1/2)*a[1]*cos(q[1]),
-x[c]/(x[c]^2+z[c]^2)^(1/2)*a[1]*cos(q[1])+(x[c]^2+z[c]^2)^(1/2),
1]);
```

$$h\_endeffector := q \rightarrow \text{vector} \left( \left[ -a_1 \sin(q_1), -\frac{z_c a_1 \cos(q_1)}{\sqrt{x_c^2 + z_c^2}}, -\frac{x_c a_1 \cos(q_1)}{\sqrt{x_c^2 + z_c^2}} + \sqrt{x_c^2 + z_c^2}, 1 \right] \right)$$

Given a vector [x,y,z,1] in camera coordinates compute the projection onto the image plane

(Standard projection equations)

```
> perspective := proc(p::vector)
vector([-1/f*p[1]/p[3], -1/f*p[2]/p[3]]):
end:
```

This is the full mapping from configuration space to image plane space

```
> h := (q1) -> vector([perspective(h_endeffector(vector([q1])))[1],
perspective(h_endeffector(vector([q1])))[2]]);
```

```
h := q1 -> vector(
[perspective(h_endeffector(vector([q1])))[1], perspective(h_endeffector(vector([q1])))[2]])
```

Let's see what this looks like symbolically

```
> h(q[1]);
```

$$\left[ \frac{a_1 \sin(q_1)}{f \left( -\frac{x_c a_1 \cos(q_1)}{\sqrt{\%1}} + \sqrt{\%1} \right)}, \frac{z_c a_1 \cos(q_1)}{f \sqrt{\%1} \left( -\frac{x_c a_1 \cos(q_1)}{\sqrt{\%1}} + \sqrt{\%1} \right)} \right]$$

$\%1 := x_c^2 + z_c^2$

## D.2 Partial Differentials

The partials of this function are as follows.

```
> diff(h(q[1])[1],q[1]); dh1dq1 := unapply(",q[1],q[2]):
```

$$\frac{a_1 \cos(q_1)}{f\left(-\frac{x_c a_1 \cos(q_1)}{\sqrt{\%1}} + \sqrt{\%1}\right)} - \frac{a_1^2 \sin(q_1)^2 x_c}{f\left(-\frac{x_c a_1 \cos(q_1)}{\sqrt{\%1}} + \sqrt{\%1}\right)^2 \sqrt{\%1}}$$

$$\%1 := x_c^2 + z_c^2$$

> diff(h(q[1])[2],q[1]); dh2dq1 := unapply(",q[1],q[2]):

$$-\frac{z_c a_1 \sin(q_1)}{f \sqrt{\%1} \left(-\frac{x_c a_1 \cos(q_1)}{\sqrt{\%1}} + \sqrt{\%1}\right)} - \frac{z_c a_1^2 \cos(q_1) x_c \sin(q_1)}{f \%1 \left(-\frac{x_c a_1 \cos(q_1)}{\sqrt{\%1}} + \sqrt{\%1}\right)^2}$$

$$\%1 := x_c^2 + z_c^2$$

> diff(h(q[1])[1],q[2]); dh1dq2 := unapply(",q[1],q[2]):

0

> diff(h(q[1])[2],q[2]); dh2dq2 := unapply(",q[1],q[2]):

0

### D.3 Dynamic Model

f is defined as the estimate for the state, given the (estimated) current state. We use a constant velocity model, with sampling interval D

> f := (xhat1,xhat2) -> vector([xhat1 + Delta\*xhat2,xhat2]);

$$f := (xhat1, xhat2) \rightarrow \text{vector}([xhat1 + \Delta xhat2, xhat2])$$

### D.4 Partial Differentials

> diff(f(q[1],q[2])[1],q[1]); df1dq1 := unapply(",q[1],q[2]):

1

> diff(f(q[1],q[2])[1],q[2]); df1dq2 := unapply(",q[1],q[2]):

$\Delta$

> diff(f(q[1],q[2])[2],q[1]); df2dq1 := unapply(",q[1],q[2]):

0

> diff(f(q[1],q[2])[2],q[2]); df2dq2 := unapply(",q[1],q[2]):

1

## D.5 Definition of Tracking Filter

partial f / partial x is a matrix of partials

> A := xhat -> matrix([ [ df1dq1(xhat[1],xhat[2]), df1dq2(xhat[1],xhat[2]) ],  
[ df2dq1(xhat[1],xhat[2]), df2dq2(xhat[1],xhat[2]) ] ]):

Check the mapping

> A(xhat);

$$\begin{bmatrix} 1 & \Delta \\ 0 & 1 \end{bmatrix}$$

partial h / partial x is a matrix of partials

> H := xhat -> matrix([ [ dh1dq1(xhat[1],xhat[2]), dh1dq2(xhat[1],xhat[2]) ],  
[ dh2dq1(xhat[1],xhat[2]), dh2dq2(xhat[1],xhat[2]) ] ]):

Check the mapping

> H(xhat);

$$\begin{bmatrix} \frac{a_1 \cos(xhat_1)}{f \%2} - \frac{a_1^2 \sin(xhat_1)^2 x_c}{f \%2^2 \sqrt{\%1}} & 0 \\ -\frac{z_c a_1 \sin(xhat_1)}{f \sqrt{\%1} \%2} - \frac{z_c a_1^2 \cos(xhat_1) x_c \sin(xhat_1)}{f \%1 \%2^2} & 0 \end{bmatrix}$$

$\%1 := x_c^2 + z_c^2$   
 $\%2 := -\frac{x_c a_1 \cos(xhat_1)}{\sqrt{\%1}} + \sqrt{\%1}$

Covariance before this step

```
> P_km1km1 := I(2):
```

Time Update - Covariance

```
> P_kkm1 := proc (xhat::vector) global H,A;
evalm(A(xhat) &* P_km1km1 &* linalg[transpose](A(xhat))):
end:
```

Time Update - State

Here is where the big win from using system dynamics comes. Some real time,  $D$ , has elapsed. We are now compensating for the time since we last observed, to when we take the next observation. Thus the linearizations below ( $f,H$ ) are done about a more accurate point. Also, if you have a local observation, it can be done about a more accurate point.

The only real difference, if we use higher-order prediction models, is that  $f$  takes more information into account. The other entries involved in computing  $f$  are usually zeroed out in  $h$  (stating that they can not be observed).

```
> xhat_kkm1 := f(xhat[1],xhat[2]):
```

Observation Update - Gain  $K_{\{k\}}$

```
> Kk := proc(xhat_kkm1::vector,Rk) global H,P_kkm1; local Hk,Hkt;
Hk := H(xhat_kkm1):
Hkt := linalg[transpose](Hk):
evalm(Hkt &* linalg[inverse](Hk &* P_kkm1 &* Hkt + Rk)):
end:
```

Observation Update - Covariance  $P_{\{k,k\}}$

```
> Pkk := (xhat_kkm1,Rk) -> evalm( P_kkm1 &* (I(1) - Kk(xhat_kkm1,Rk)&*H(xhat_kkm1))):
```

Observation Update - State Here,  $z_k$  is the observation back from the (physical) system.

```
> xhat_kk := xhat_kkm1 + Kk&*(zk - h(xhat_kkm1)):
```

Increment  $k$  by one, and iterate the filter.

## APPENDIX E

### CASE 5: 3DOF PUMA ROBOTIC ARM

This appendix presents an annotated Maple session. This session derives the observation equation  $\mathbf{h}_k(\mathbf{x}_k)$  and dynamic model equation  $\mathbf{f}_k(\mathbf{x}_k)$  for the case presented. In the interest of space, the equations for the partial derivatives of these functions are omitted.

Case 4 - 3DOF PUMA arm

Constant Velocity Motion Model

Perspective Projection Imaging Model

```
> digits := 8: # to match C 'double precision'
> with(linalg): interface(labelling=false): readlib(C):

These vectors are the DH parameters of the arm. (Here, it's the PUMA).
> a := array(1..6, [0, 431.8, -20.32, 0, 0, 0]):
d := array(1..6, [0, 149.09, 0, 433.07, 0, 56.25]):
alpha := array(1..6, [-Pi/2, 0, Pi/2, -Pi/2, Pi/2, 0]):
#fx := 1000: fy := 1185:
#cameraheight := 0000:
#cameradepth := 5000:
cameradepth := 'cameradepth': cameraheight := 'cameraheight':
fx := 'focallengthx': fy := 'focallengthy':
```



## E.1 Projection Equations

This is the standard homogeneous transformation, given the DH parameters (see, e.g. Spong P65-66)

```
> A := proc(theta::array,i::integer) global a,d,alpha;
array(1..4,1..4,[
[cos(theta[i]), -sin(theta[i])*cos(alpha[i]),
sin(theta[i])*sin(alpha[i]), a[i]*cos(theta[i])],
[sin(theta[i]), cos(theta[i])*cos(alpha[i]),
-cos(theta[i])*sin(alpha[i]), a[i]*sin(theta[i])],
[0, sin(alpha[i]), cos(alpha[i]), d[i]],
[0, 0, 0, 1]]);
end:
```

Given a vector  $[x,y,z,1]$  in camera coordinates compute the projection onto the image plane.

```
> perspective := proc(p::vector) vector([fx*p[1]/p[3],fy*p[2]/p[3]]): end:
```

Define the transformation from the world coordinates to the camera coordinates.

Here, the camera is at +5000 on the y-axis, with +z of the camera pointing at the z0 axis, at height 1000

```
> camera := matrix([[ -1,0,0,0],[0,0,-1,cameraheight],[0,-1,0,cameradepth],[0,0,0,1]]);
```

$$camera := \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & cameraheight \\ 0 & -1 & 0 & cameradepth \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Define mapping from end-effector coordinates to image plane coordinates.

```
> H3 := (q,p) -> perspective(evalm(camera&*A(q,1)&*A(q,2)&*A(q,3)&*p)):
```

```
> H2 := (q,p) -> perspective(evalm(camera&*A(q,1)&*A(q,2)&*p)):
```

```
> H1 := (q,p) -> perspective(evalm(camera&*A(q,1)&*p)):
> H0 := (q,p) -> perspective(evalm(camera&*p)):
```

This defines a few positions in workspace and configuration space

```
> q := array(1..6): p := array(1..4): straightarm := array([-1040,-20,150,1]):
# straight arm in upright state (in q1 frame) q_left := array([-Pi,-Pi,Pi/2,0,0,0]):
q_center := array([-Pi/2,-Pi,Pi/2,0,0,0]): q_right := array([0,-Pi,Pi/2,0,0,0]):
```

location in the image of a point in coordinate system 0 (slightly boring)

```
> H0(q,p);
```

$$\left[ -\frac{\text{focallength}x p_1}{-p_2 + \text{cameradepth} p_4}, \frac{\text{focallength}y (-p_3 + \text{cameraheight} p_4)}{-p_2 + \text{cameradepth} p_4} \right]$$

location in the image of a point in coordinate system 1

```
> H1(q,p);
```

$$\left[ \frac{\text{focallength}x (-\cos(q_1) p_1 + \sin(q_1) p_3)}{-\sin(q_1) p_1 - \cos(q_1) p_3 + \text{cameradepth} p_4}, \frac{\text{focallength}y (p_2 + \text{cameraheight} p_4)}{-\sin(q_1) p_1 - \cos(q_1) p_3 + \text{cameradepth} p_4} \right]$$

location in the image of a point in coordinate system 2

```
> H2(q,p);
```

$$\left[ \text{focallength}x (-\cos(q_1) \cos(q_2) p_1 + \cos(q_1) \sin(q_2) p_2 + \sin(q_1) p_3 \right. \\ \left. + (-431.8 \cos(q_1) \cos(q_2) + 149.09 \sin(q_1)) p_4) / (-\sin(q_1) \cos(q_2) p_1 + \sin(q_1) \sin(q_2) p_2 \right. \\ \left. - \cos(q_1) p_3 + (-431.8 \sin(q_1) \cos(q_2) - 149.09 \cos(q_1) + \text{cameradepth} p_4) p_4), \text{focallength}y \right. \\ \left. (\sin(q_2) p_1 + \cos(q_2) p_2 + (431.8 \sin(q_2) + \text{cameraheight} p_4) p_4) / (-\sin(q_1) \cos(q_2) p_1 \right. \\ \left. + \sin(q_1) \sin(q_2) p_2 - \cos(q_1) p_3 \right. \\ \left. + (-431.8 \sin(q_1) \cos(q_2) - 149.09 \cos(q_1) + \text{cameradepth} p_4) p_4) \right]$$

location in the image of a point in coordinate system 3

```
> H3(q,p);
```

$$\begin{aligned}
& \left[ \text{focallengthx} \left( (-\cos(q_1) \cos(q_2) \cos(q_3) + \cos(q_1) \sin(q_2) \sin(q_3)) p_1 + \sin(q_1) p_2 \right. \right. \\
& \quad + (-\cos(q_1) \cos(q_2) \sin(q_3) - \cos(q_1) \sin(q_2) \cos(q_3)) p_3 + (20.32 \cos(q_1) \cos(q_2) \cos(q_3) \\
& \quad - 20.32 \cos(q_1) \sin(q_2) \sin(q_3) - 431.8 \cos(q_1) \cos(q_2) + 149.09 \sin(q_1)) p_4 \left. \right) / \left( \right. \\
& \quad (-\sin(q_1) \cos(q_2) \cos(q_3) + \sin(q_1) \sin(q_2) \sin(q_3)) p_1 - \cos(q_1) p_2 \\
& \quad + (-\sin(q_1) \cos(q_2) \sin(q_3) - \sin(q_1) \sin(q_2) \cos(q_3)) p_3 + (20.32 \sin(q_1) \cos(q_2) \cos(q_3) \\
& \quad - 20.32 \sin(q_1) \sin(q_2) \sin(q_3) - 431.8 \sin(q_1) \cos(q_2) - 149.09 \cos(q_1) + \text{cameradepth}) p_4 \\
& \left. \right), \text{focallengthy} \left( (\sin(q_2) \cos(q_3) + \cos(q_2) \sin(q_3)) p_1 + (\sin(q_2) \sin(q_3) - \cos(q_2) \cos(q_3)) p_3 \right. \\
& \quad + (-20.32 \sin(q_2) \cos(q_3) - 20.32 \cos(q_2) \sin(q_3) + 431.8 \sin(q_2) + \text{cameraheight}) p_4 \left. \right) / \left( \right. \\
& \quad (-\sin(q_1) \cos(q_2) \cos(q_3) + \sin(q_1) \sin(q_2) \sin(q_3)) p_1 - \cos(q_1) p_2 \\
& \quad + (-\sin(q_1) \cos(q_2) \sin(q_3) - \sin(q_1) \sin(q_2) \cos(q_3)) p_3 + (20.32 \sin(q_1) \cos(q_2) \cos(q_3) \\
& \quad - 20.32 \sin(q_1) \sin(q_2) \sin(q_3) - 431.8 \sin(q_1) \cos(q_2) - 149.09 \cos(q_1) + \text{cameradepth}) p_4 \\
& \left. \right) \left. \right]
\end{aligned}$$

## E.2 Dynamic Model

$f$  is defined as the estimate for the state, given the (estimated) current state. We use a constant velocity model, with sampling interval  $\Delta$  and  $\hat{x}_1$  and  $\hat{x}_2$  are the joint angle estimates and joint angle velocities, respectively

```
> f := (xhat1,xhat2) -> vector([xhat1 + Delta*xhat2,xhat2]);
```

$$f := (xhat1, xhat2) \rightarrow \text{vector}([xhat1 + \Delta xhat2, xhat2])$$

## E.3 Partial Differentials

```
> diff(f(q[1],q[2])[1],q[1]); df1dq1 := unapply(",q[1],q[2]):
```

1

```
> diff(f(q[1],q[2])[1],q[2]); df1dq2 := unapply(",q[1],q[2]):
```

$\Delta$

```
> diff(f(q[1],q[2])[2],q[1]); df2dq1 := unapply(",q[1],q[2]):
```

0

```
> diff(f(q[1],q[2])[2],q[2]); df2dq2 := unapply(",q[1],q[2]):
```

1

## APPENDIX F

### CASE 6: INCOMPLETE OBJECT MODEL

This appendix presents an annotated Maple session. This session derives the observation equation  $\mathbf{h}_k(\mathbf{x}_k)$  and dynamic model equation  $\mathbf{f}_k(\mathbf{x}_k)$  for the case presented. In the interest of space, the equations for the partial derivatives of these functions are omitted.

Case 6 - Incomplete Motion Model

2DOF Arm

Unknown Link Lengths

Constant Position Motion Model

```
> digits := 8: # to match C 'double precision'  
> with(linalg): interface(labelling=false): readlib(C):
```

These vectors are the DH parameters of the arm. (Here, it's a 2d planar arm).

```
> a := array(1..2):  
d := array(1..2, [0,0]):  
alpha := array(1..6, [0,0]):  
scalex := 1: scaley := 1:  
#camerax := 512/2: cameray := 485/2: cameradepth := 0:  
camerax := 'camerax': cameray := 'cameray': cameradepth :=  
'cameradepth':
```

## F.1 Projection Equations

This is the standard homogeneous transformation, given the DH parameters (see, e.g. Spong P65-66)

```
> A := proc(theta::array,a::array,i::integer) global d,alpha;
array(1..4,1..4,[
[cos(theta[i]), -sin(theta[i])*cos(alpha[i]),
sin(theta[i])*sin(alpha[i]), a[i]*cos(theta[i])],
[sin(theta[i]), cos(theta[i])*cos(alpha[i]),
-cos(theta[i])*sin(alpha[i]), a[i]*sin(theta[i])],
[0, sin(alpha[i]), cos(alpha[i]), d[i]],
[0, 0, 0, 1]]);
end:
```

Given a vector  $[x,y,z,1]$  in camera coordinates compute the projection onto the image plane.

```
> perspective := proc(p::vector) vector([fx*p[1]/p[3],fy*p[2]/p[3]]): end:
ortho := proc(p::vector) vector([scalex*p[1],scaley*p[2]]): end:
```

Define the transformation from the world coordinates to the camera coordinates.

```
> camera := matrix([[1,0,0,camerax],[0,1,0,cameray],[0,0,1,cameradepth],[0,0,0,1]]);
```

$$camera := \begin{bmatrix} 1 & 0 & 0 & camera_x \\ 0 & 1 & 0 & camera_y \\ 0 & 0 & 1 & camera_{depth} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Define mapping from end-effector coordinates to image plane coordinates.

```
> H2 := (x,p) -> ortho(evalm(camera&*
A(array([x[1],x[2]]),array([x[3],x[4]]),1)&*
A(array([x[1],x[2]]),array([x[3],x[4]]),2)&*p)):
```

```
> H1 := (x,p) -> ortho(evalm(camera&* A(array([x[1],x[2]]),array([x[3],x[4]]),1)&*p)):
> H0 := (x,p) -> ortho(evalm(camera&* p)):
```

This defines a few positions in workspace and configuration space

```
> x := array([theta[1],theta[2],a[1],a[2]]): p := array(1..4): p[4] := 1:
x := array(1..4): p := array(1..4):
```

location in the image of a point in coordinate system 0 (slightly boring)

```
> H0(x,p);
```

$$[p_1 + camera_x p_4, p_2 + camera_y p_4]$$

location in the image of a point in coordinate system 1

```
> H1(x,p);
```

$$\begin{bmatrix} \cos(x_1) p_1 - \sin(x_1) p_2 + (x_3 \cos(x_1) + camera_x) p_4, \\ \sin(x_1) p_1 + \cos(x_1) p_2 + (x_3 \sin(x_1) + camera_y) p_4 \end{bmatrix}$$

location in the image of a point in cs 2

```
> H2(x,p);
```

$$\begin{bmatrix} (\cos(x_1) \cos(x_2) - \sin(x_1) \sin(x_2)) p_1 + (-\cos(x_1) \sin(x_2) - \sin(x_1) \cos(x_2)) p_2 \\ + (\cos(x_1) x_4 \cos(x_2) - \sin(x_1) x_4 \sin(x_2) + x_3 \cos(x_1) + camera_x) p_4, \\ (\sin(x_1) \cos(x_2) + \cos(x_1) \sin(x_2)) p_1 + (\cos(x_1) \cos(x_2) - \sin(x_1) \sin(x_2)) p_2 \\ + (\sin(x_1) x_4 \cos(x_2) + \cos(x_1) x_4 \sin(x_2) + x_3 \sin(x_1) + camera_y) p_4 \end{bmatrix}$$

## F.2 Dynamic Model

$f$  is defined as the estimate for the state, given the (estimated) current state. We use a constant velocity model, with sampling interval  $D$   $\hat{x}1$  and  $\hat{x}2$  are the joint angle estimates and joint angle velocities, respectively

```
> f := (xhat1,xhat2) -> vector([xhat1 + Delta*xhat2,xhat2]);
```

$$f := (xhat1, xhat2) \rightarrow \text{vector}([xhat1 + \Delta xhat2, xhat2])$$

### F.3 Partial Differentials

```
> diff(f(q[1],q[2])[1],q[1]); df1dq1 := unapply(",q[1],q[2]):
```

1

```
> diff(f(q[1],q[2])[1],q[2]); df1dq2 := unapply(",q[1],q[2]):
```

$\Delta$

```
> diff(f(q[1],q[2])[2],q[1]); df2dq1 := unapply(",q[1],q[2]):
```

0

```
> diff(f(q[1],q[2])[2],q[2]); df2dq2 := unapply(",q[1],q[2]):
```

1



## REFERENCES

- [1] Y. Azoz, L. Devi, and R. Sharma, "Human arm tracking using multimodal localization and constraint fusion," in *Proceedings 1997 Advanced Display Federated Laboratory Symposium*, 1997.
- [2] K. Akita, "Image sequence analysis of real world human motion," *Pattern Recognition*, vol. 17, no. 1, pp. 73–83, 1984.
- [3] W. J. Wilson, C. C. W. Hulls, and G. S. Bell, "Relative end-effector control using Cartesian position based visual servoing," *IEEE Transactions on Robotics and Automation*, vol. 12, pp. 684–696, Oct. 1996.
- [4] P. K. Allen, A. Timcenko, B. Yoshimi, and P. Michelman, "Automated tracking and grasping of a moving object with a robotic hand-eye system," *IEEE Transactions on Robotics and Automation*, vol. 9, pp. 152–165, Apr. 1993.
- [5] P. Matteucci, C. S. Regazzoni, and G. L. Foresti, "Real-time approach to 3-D object tracking in complex scenes," *Electronics Letters*, vol. 30, pp. 475–477, Mar. 1994.
- [6] Y. Shirai, Y. Mae, and S. Yamamoto, "Object tracking by using optical flows and edges," in *Robotics Research - The Seventh International Symposium*, Berlin: Springer-Verlag, 1996, pp. 440–447.
- [7] L. Baumela and D. Maravall, "Real-time target tracking," *IEEE Aerospace and Electronic Systems Magazine*, vol. 10, pp. 4–7, July 1995.
- [8] S. Krishnan and D. Raviv, "2D feature tracking algorithm for motion analysis," *Pattern Recognition*, vol. 28, pp. 1103–1126, Aug. 1995.
- [9] J. K. Aggarwal and N. Nandhakumar, "On the computation of motion from sequences of images - A review," *Proceedings of the IEEE*, vol. 76, pp. 917–935, Aug. 1988.
- [10] D. B. Gennery, "Visual tracking of known three-dimensional objects," *International Journal of Computer Vision*, vol. 7, no. 3, pp. 243–270, 1992.
- [11] R. Lopez, A. Colmenarez, and T. S. Huang, "Vision-based head and facial feature tracking," in *Advanced Displays and Interactive Displays Federated Laboratory Consortium, Annual Symposium*, Advanced Displays and Interactive Displays Federated Laboratory Consortium, Jan. 1997.
- [12] N. P. Papanikolopoulos, "Selection of features and evaluation of visual measurements during robotic visual servoing tasks," *Journal of Intelligent and Robotic Systems*, vol. 13, pp. 279–304, July 1995.

- [13] G. Young and R. Chellappa, "3D motion estimation using a sequence of noisy stereo images: Models, estimation, and uniqueness results," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, pp. 735–759, Aug. 1990.
- [14] A. Singh and P. Allen, "Image flow computation: An estimation-theoretic framework and a unified perspective," *Computer Vision Graphics and Image Processing: Image Understanding*, vol. 56, pp. 152–177, Sept. 1992.
- [15] P. Anandan, "A computational framework and an algorithm for the measurement of visual motion," *International Journal of Computer Vision*, vol. 2, pp. 283–310, Jan. 1989.
- [16] H. Liu, T. Hong, M. Herman, and R. Chellappa, "A general motion model and spatio-temporal filters for computing optical flow," *International Journal of Computer Vision*, vol. 22, no. 2, pp. 141–172, 1997.
- [17] A. Singh, "An estimation-theoretic framework for image-flow computation," in *Proceedings International Conference Computer Vision*, 1990, pp. 168–177.
- [18] B. K. P. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, pp. 185–203, Aug. 1982.
- [19] E. H. Adelson and J. R. Bergen, "Spatiotemporal energy models for the perception of motion," *Journal of the Optic Society of America*, vol. 2, pp. 284–298, Feb. 1985.
- [20] D. Heeger, "Optical flow using spatiotemporal filters," *International Journal of Computer Vision*, vol. 1, no. 4, pp. 279–302, 1987.
- [21] D. J. Fleet and A. D. Jepson, "Computation of component image velocity from local phase information," *International Journal of Computer Vision*, vol. 5, no. 1, pp. 77–104, 1990.
- [22] J. Weber and J. Malik, "Robust computation of optical flow in a multi-scale differential framework," *International Journal of Computer Vision*, vol. 14, pp. 67–81, Jan. 1995.
- [23] B. K. P. Horn, *Robot Vision*. New York: McGraw-Hill, 1986.
- [24] G. Adiv, "Determining three-dimensional motion and structure from optical flow generated by several moving objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 7, pp. 384–401, July 1985.
- [25] J. R. Bergen, P. J. Burt, R. Hingorani, and S. Peleg, "Computing two motions from three frames," in *Proceedings International Conference Computer Vision*, 1990.
- [26] D. Lowe, "Robust model-based motion tracking through the integration of search and estimation," *International Journal of Computer Vision*, vol. 8, pp. 113–122, Aug. 1992.
- [27] G. Hager, "Real-time feature tracking and projective invariance as a basis for hand-eye coordination," in *Proceedings IEEE Computer Society Conference on Computer Vision Pattern Recognition*, 1994, pp. 533–539.
- [28] P. I. Corke, "Visual control of robot manipulators - A review," in *Visual Servoing: Real-Time Control of Robot Manipulators Based on Visual Sensory Feedback*, K. Hashimoto, Ed., Singapore: World Scientific, Inc, 1993, pp. 1–32.
- [29] J. Shi and C. Tomisito, "Good features to track," in *Proceedings IEEE Computer Society Conference on Computer Vision Pattern Recognition*, June 1994, pp. 593–600.

- [30] C. Bregler and J. Malik, "Video motion capture," in *Proceedings SIGGRAPH 1998*, 1998.
- [31] J. C. Clarke and A. Zisserman, "Detection and tracking of independent motion," *Image and Vision Computing*, vol. 14, pp. 565–572, Aug. 1996.
- [32] R. Deriche and O. Faugeras, "Tracking line segments," *Image and Vision Computing*, vol. 8, pp. 261–270, Nov. 1991.
- [33] K. J. Bradshaw, I. D. Reid, and D. W. Murray, "The active recovery of 3D motion trajectories and their use in prediction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, pp. 219–234, Mar. 1997.
- [34] J. Denzler and H. Niemann, "Combination of simple vision modules for robust real-time motion tracking," *European Transactions on Telecommunications and Related Technologies*, vol. 6, pp. 275–286, May/June 1995.
- [35] A. Rizzi and D. E. Koditschek, "Further progress in robot juggling: The spatial two-juggle," in *Proceedings IEEE International Conference Robotics and Automation*, 1993, pp. 919–924.
- [36] M. Ishii, S. Sakane, M. Kakikura, and Y. Mikami, "A 3D sensor system for teaching robot paths and environments," *International Journal of Robotics Research*, vol. 6, no. 2, pp. 45–59, 1987.
- [37] R. S. Stephens, "Real-time 3D object tracking," *Image and Vision Computing*, vol. 8, pp. 91–96, Feb. 1990.
- [38] A. J. Bray, "Tracking objects using image disparities," *Image and Vision Computing*, vol. 8, pp. 4–9, Feb. 1990.
- [39] A. Kosaka and A. C. Kak, "Fast vision-guided robot navigation using model-based reasoning and prediction of uncertainties," *Computer Vision and Image Understanding*, vol. 56, pp. 271–329, Nov. 1992.
- [40] R. Brunelli and T. Poggio, "Face recognition: Features versus templates," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, pp. 1042–1052, Oct. 1993.
- [41] H. Li, P. Roivainen, and R. Forcheimer, "3-D motion estimation in model-based facial image coding," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, pp. 545–555, June 1993.
- [42] L. Tang, "Human face modeling, analysis, and synthesis," Ph.D. dissertation, University of Illinois, Urbana, IL, 1996.
- [43] P. J. Burt, C. Yen, and X. Xu, "Local correlation measures for motion analysis: A comparative study," in *Proceedings IEEE Conference Pattern Recognition Image Processing*, 1982, pp. 269–274.
- [44] A. Rosenfeld and A. Kak, *Digital Picture Processing*, 2nd ed. Cambridge, MA: Academic Press, 1982.
- [45] R. Brunelli and S. Messelodi, "Robust estimation of correlation with applications to computer vision," *Pattern Recognition*, vol. 28, no. 6, pp. 833–841, 1995.
- [46] F. R. Hampel, P. J. Rousseeuw, E. M. Ronchetti, and W. A. Stahel, *Robust Statistics: The approach based on influence functions*. Chichester, UK: John Wiley & Sons, 1986.

- [47] P. J. Huber, "Robust estimation of a location parameter," *Annals of Mathematical Statistics*, vol. 35, pp. 73–101, 1964.
- [48] P. J. Huber, *Robust Statistical Procedures*, 2nd ed. Philadelphia PA: Society for Industrial and Applied Mathematics, 1996.
- [49] R. Brunelli and T. Poggio, "Template matching: Matched spatial filters and beyond," *Pattern Recognition*, vol. 30, no. 5, pp. 751–768, 1997.
- [50] G. Hager and P. Belhumeur, "Real-time tracking of image regions with changes in geometry and illumination," in *Proceedings IEEE Computer Society Conference on Computer Vision Pattern Recognition*, 1996, pp. 403–410.
- [51] D. Slater and G. Healey, "Using a spectral reflectance model for the illumination-invariant recognition of local image structure," in *Proceedings IEEE Computer Society Conference on Computer Vision Pattern Recognition*, 1996, pp. 770–775.
- [52] S. Birchfield and C. Tomasi, "A pixel dissimilarity measure that is insensitive to image sampling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, pp. 401–406, Apr. 1998.
- [53] P. Anandan, "Measuring visual motion from image sequences," COINS Department, University of Massachusetts, Tech. Rep. COINS-TR-87-21, 1987.
- [54] L. Matthies, T. Kanade, and R. Szeliski, "Kalman filter-based algorithms for estimating depth from image sequences," *International Journal of Computer Vision*, vol. 3, pp. 209–236, 1989.
- [55] T. Tommasini, A. Fusiello, E. Trucco, and V. Roberto, "Making good features to track better," in *Proceedings IEEE Computer Society Conference on Computer Vision Pattern Recognition*, 1998, pp. 145–149.
- [56] H. H. Nagel, "Overview on image sequence analysis," in *Image Sequence Processing and Dynamic Scene Analysis*, T. S. Huang, Ed., Berlin: Springer-Verlag, 1983, pp. 2–38.
- [57] K. Hashimoto, A. Aoki, and T. Noritsugu, "Visual servoing with redundant features," in *Proceedings of the Conference on Decision and Control*, Dec. 1996, pp. 2483–2483.
- [58] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering, Transactions ASME, Series D*, vol. 82, pp. 35–45, Mar. 1960.
- [59] J. C. Willems, "Dynamical systems, controllability, and observability: A post-modern point of view," in *Mathematical System Theory: The Influence of R. E. Kalman*, A. C. Antoulas, Ed., Berlin: Springer-Verlag, 1991, pp. 17–39.
- [60] Y. Bar-Shalom and T. E. Fortmann, *Tracking and Data Association*. Orlando, FL: Academic Press, 1988.
- [61] R. E. Kalman and R. S. Bucy, "New results in linear filtering and prediction theory," *Journal of Basic Engineering, Transactions ASME, Series D*, vol. 83, pp. 95–108, Mar. 1961.
- [62] T. Kailath, "From Kalman filtering to innovations, martingales, scattering and other nice things," in *Mathematical System Theory: The Influence of R. E. Kalman*, A. C. Antoulas, Ed., Berlin: Springer-Verlag, 1991, pp. 55–88.

- [63] J. Hallam, "Resolving observer motion by object tracking," in *Proceedings International Joint Conference on Artificial Intelligence*, 1983, pp. 792–798.
- [64] A. R. Morley and A. S. Wilsdon, "Multiradar tracking in a multisite environment," in *Proceedings RADAR-77 IEE International Conference*, 1977, p. 66.
- [65] N. Ayache and O. D. Faugeras, "HYPER: A new approach for the recognition and positioning of two-dimensional objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, pp. 44–54, Jan. 1986.
- [66] P. H. G. Coelho and L. F. A. Nunes, "Application of Kalman filtering to robot manipulators," in *Recent Trends in Robotics: Modeling, Control, and Education*, M. Jamshidi, L. Y. S. Luh, and M. Shahinpoor, Eds., New York: Elsevier Science, 1986, pp. 35–40.
- [67] D. B. Gennery, "Tracking known three-dimensional objects," in *Proceedings American Association Artificial Intelligence*, 1982, pp. 13–17.
- [68] A. Z. Meri, "On monocular perspective of 3D moving objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 2, pp. 582–583, Nov. 1980.
- [69] J. Q. Fang and T. S. Huang, "Some experiments on estimating the 3D motion parameters of a rigid body from two consecutive image frames," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, pp. 545–554, 1984.
- [70] D. P. Huttenlocher, J. J. Noh, and W. J. Rucklidge, "Tracking non-rigid objects in complex scenes," in *Proceedings International Conference Computer Vision*, Apr. 1993, pp. 93–101.
- [71] A. Gee and R. Cipolla, "Fast visual tracking by temporal consensus," *Image and Vision Computing*, vol. 14, pp. 105–114, 1996.
- [72] O. D. Faugeras, N. Ayache, and B. Faverjon, "Building visual maps by combining noisy stereo measurements," in *Proceedings IEEE International Conference Robotics and Automation*, 1986, pp. 1433–1438.
- [73] T. J. Broida and R. Chellappa, "Estimation of object motion parameters from noise images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, pp. 90–99, Jan. 1986.
- [74] L. Matthies and S. A. Shafer, "Error modeling in stereo navigation," *IEEE Journal of Robotics and Automation*, vol. RA-3, pp. 239–248, June 1987.
- [75] E. Dickmanns and V. Graefe, "Dynamic monocular machine vision," *Machine Vision and Applications*, vol. 1, pp. 223–240, Apr. 1988.
- [76] E. Dickmanns and V. Graefe, "Applications of dynamic monocular machine vision," *Machine Vision and Applications*, vol. 1, pp. 241–261, Apr. 1988.
- [77] J. J. Wu, R. E. Rink, T. M. Caelli, and V. G. Gourishankar, "Recovery of the 3D location and motion of a rigid object through camera image (an extended Kalman filter approach)," *International Journal of Computer Vision*, vol. 4, pp. 373–394, Apr. 1988.
- [78] N. A. Andersen, O. Ravn, and A. T. Sorensen, "Real-time vision based control of servomechanical systems," in *Proceedings of the 2nd International Symposium on Experimental Robotics*, June 1991.

- [79] J. W. Lee, M. S. Kim, and I. S. Kweon, "A Kalman filter based visual tracking algorithm for an object moving in 3D," in *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1995, pp. 342–347.
- [80] P. Wunsch and G. Hirzinger, "Real-time visual tracking of 3D objects with dynamic handling of occlusion," in *Proceedings IEEE International Conference Robotics and Automation*, 1997, pp. 2868–2873.
- [81] D. Metaxas and D. Terzopoulos, "Shape and nonrigid motion estimation through physics-based synthesis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, pp. 580–591, June 1993.
- [82] A. Blake, R. Curwen, and A. Zisserman, "A framework for spatiotemporal control in the tracking of visual contours," *International Journal of Computer Vision*, vol. 11, pp. 127–145, Oct. 1993.
- [83] K. Stark and S. Fuchs, "A method for tracking the pose of known 3D objects based on an active contour model," in *Proceedings International Conference Pattern Recognition*, 1996, pp. 905–909.
- [84] C. K. Chui and G. Chen, *Kalman Filtering with Real-Time Applications*. Berlin: Springer-Verlag, 1987.
- [85] J. L. Goldberg, *Matrix Theory with Applications*. New York: McGraw-Hill, 1991.
- [86] C. K. Chui and G. Chen, *Linear Systems and Optimal Control*. Berlin: Springer-Verlag, 1989.
- [87] F. L. Lewis, *Optimal Estimation with an Introduction to Stochastic Control Theory*. New York: John Wiley & Sons, 1986.
- [88] R. G. Brown, *Introduction to Random Signal Analysis and Kalman Filtering*. New York: John Wiley & Sons, 1983.
- [89] P. S. Maybeck, *Stochastic Models, Estimation, and Control*. New York: Academic Press, Inc, 1979.
- [90] G. J. Bierman, "Numerical experience with modern estimation algorithms," in *Proceedings of the Conference on Decision and Control*, Dec. 1985, pp. 1896–1901.
- [91] Z. Zhang and O. Faugeras, *3D Dynamic Scene Analysis*. Berlin: Springer-Verlag, 1992.
- [92] W. J. Rey, *Introduction to Robust and Quasi-Robust Statistical Methods*. Berlin: Springer, 1983.
- [93] M. Spong and M. Vidyasagar, *Robot Dynamics and Control*. New York: John Wiley & Sons, 1989.
- [94] A. Hauck, S. Lanser, and C. Zierl, "Hierarchical recognition of articulated objects from single perspective views," in *Proceedings IEEE Computer Society Conference on Computer Vision Pattern Recognition*, 1997, pp. 870–883.
- [95] J. E. Lloyd, J. S. Beis, D. K. Pai, and D. G. Lowe, "Model-based telerobotics with vision," in *Proceedings IEEE International Conference Robotics and Automation*, vol. 2, Apr. 1997, pp. 1297–1304.

- [96] H. A. Rowley and J. M. Rehg, "Analyzing articulated motion using expectation-maximization," in *Proceedings IEEE Computer Society Conference on Computer Vision Pattern Recognition*, 1997, pp. 935–941.
- [97] J. C. Latombe, *Robot Motion Planning*. Norwell, MA: Kluwer Academic, 1991.
- [98] M. Woo, J. Neider, and T. Davis, *The OpenGL Programming Guide*. Reading, MA: Addison-Wesley, 1996.
- [99] A. Zolghadri, "An algorithm for real-time failure detection in Kalman filters," *IEEE Transactions on Automatic Control*, vol. 41, pp. 1537–1539, Oct. 1996.
- [100] R. Horaud, B. Canio, and O. Leboulloux, "An analytic solution for the perspective 4-point problem," *Computer Vision Graphics and Image Processing*, vol. 47, no. 1, pp. 33–44, 1989.
- [101] R. M. Haralick and L. G. Shapiro, *Computer and Robot Vision*, Vol. 2. Reading, MA: Addison-Wesley, 1993.
- [102] C. Chen, *Linear System Theory and Design*. Fort Worth, TX: Harcourt Brace, 1984.
- [103] S. Hutchinson, G. Hager, and P. Corke, "A tutorial on visual servo control," *IEEE Transactions on Robotics and Automation*, vol. 12, pp. 651–669, Oct. 1996.
- [104] A. Watt, *3-D Computer Graphics*, 2nd ed. Reading, MA: Addison-Wesley, 1995.
- [105] B. J. Nelson and P. K. Khosla, "Strategies for increasing the tracking region of an eye-in-hand system by singularity and joint limit avoidance," *International Journal of Robotics Research*, vol. 14, pp. 255–269, June 1995.
- [106] K. Nickels and S. Hutchinson, "Integrated object models for robust visual tracking," in *Proceedings of Workshop on Robust Vision for Vision-based Control of Motion*, 1998.
- [107] K. Nickels and S. Hutchinson, "Weighting observations: The use of kinematic models in object tracking," in *Proceedings IEEE International Conference Robotics and Automation*, 1998.
- [108] K. Nickels and S. Hutchinson, "Characterizing the uncertainties in point feature motion for model-based object tracking," in *Proceedings of Workshop on New Trends in Image-Based Robot Servicing, Grenoble, France*, 1997, pp. 53–63.

## VITA

Kevin Michael Nickels received a B.S. degree with distinction in computer and electrical engineering in 1993 from Purdue University and an M.S. degree in electrical engineering in 1996 from the University of Illinois at Urbana-Champaign.

He is a member of Eta Kappa Nu, Tau Beta Phi, the Institute of Electrical and Electronics Engineers, and Phi Kappa Phi. His published articles include “Textured Image Segmentation: Returning Multiple Solutions,” by Kevin Nickels and Seth Hutchinson in *Image and Vision Computing*, Vol. 15, 1997, pp. 781-795; “Weighting Observations: The use of Kinematic Models in Object Tracking,” by Kevin Nickels and Seth Hutchinson in *Proceedings of the 1998 IEEE International Conference on Robotics and Automation*; “Characterizing the Uncertainties in Point Feature Motion for Model-Based Object Tracking,” by K. Nickels and S. Hutchinson in *Proceedings 1998 Workshop on New Trends in Image-Based Robot Servoing*; and “Integrated Object Models for Robust Visual Tracking,” by Kevin Nickels and Seth Hutchinson in *Proceedings 1998 Workshop on Robust Vision for Vision-Based Control of Motion*.

While at the University of Illinois at Urbana-Champaign, he taught the Microcomputer Laboratory (ECE311) for three semesters as a teaching assistant, and one semester as an instructor.