1

MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS
STANDARD REFERENCE MATERIAL 1010a
(ANSI and ISO TEST CHART No. 2)

National Library
of Canada

Bibliothèque nationale
du Canada

Canadian Theses Service

Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

**Motion And Structure Estimation
In Noisy Monocular Image Sequences:
Specialization To Planar Motion**

by

Victor C. Aitken

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfilment of
the requirements for the degree of
Master of Engineering in Electrical Engineering
Department of Systems and Computer Engineering

Carleton University
Ottawa, Ontario
April 12th, 1991

© Copyright
1991, Victor C. Aitken.

National Library
of Canada

Bibliothèque nationale
du Canada

Canadian Theses Service Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-315-68897-1

Canada

The undersigned recommend to
The Faculty of Graduate Studies and Research
acceptance of the thesis

**Motion And Structure Estimation:
In Noisy Monocular Image Sequences:
Specialization To Planar Motion.**

submitted by
Victor C. Aitken, B.A.Sc.
in partial fulfilment
of the requirements for the degree of
Master of Engineering in Electrical Engineering

Thesis Supervisor

Chairman, Department of Systems and Computer Engineering

Carleton University

April 12, 1991

Abstract

Object motion and structure estimation from noisy measurements of feature positions in monocular image sequences is investigated with emphasis on the case in which objects of interest are constrained to lie on a known planar surface. Linear motion and structural models, and a non-linear measurement model are proposed for planar motion observed from an arbitrary position on or above the known plane. An iterated extended Kalman filter is applied, with measurement noise levels from 5-20% of the object image size and feature point occlusion due to object motion, to refine estimates of motion and structural parameters over an arbitrary number of image frames. The case considered here is that of a rigid object undergoing constant translational and angular velocity for which image feature position measurements and correspondence information are available. One feature point is assumed to lie at a known or approximated height above the plane of motion. Good performance of the proposed technique is demonstrated with simulated imagery through comparison of Monte Carlo simulation results with approximate Cramer-Rao bounds and results obtained from Broida and Chellappa's methods [IEEE Trans. Pattern Anal. Mach. Intell. PAMI-8, 1986, p90; IEEE Trans. Aerosp. Electron. Syst. Vol. 26, No. 4, 1990, p639].

Acknowledgements

I would first like to thank my supervisor, Dr. H. M. Schwartz, who has been both encouraging and demanding, and whose high standards have served as an example of academic and moral excellence. I would also like to thank Dr. A.U. Sheikh who provided academic counseling and guidance throughout my academic program. The students and faculty at Carleton have been a constant source of intellectual stimulation. I would especially like to thank Richard Gourdeau for taking the time to discuss my work on numerous occasions and for his invaluable help and suggestions.

I gratefully acknowledge financial assistance provided by the Natural Sciences and Engineering Research Council of Canada through their graduate scholarship program, and by the Canadian Department of National Defence. Management and my colleagues at the Defence Research Establishment Suffield have been very supportive of my academic program. In particular, I want to thank Dr. J. Molton, Dr. R. Suart, and Robert Chesney for supporting my proposals for education leave, providing ongoing support and advice during the academic program, and contributing to a work environment that encourages and fosters academic achievement.

Finally, I would like to thank my wife, Maria, and our two children, Kathleen and Julie, for their patience, understanding, and support.

Contents

1	Introduction	1
1.1	Two Fundamental Approaches to Motion Analysis	3
1.2	Review of Motion Analysis Through Extended Kalman Filtering . . .	5
1.3	Thesis Statement	12
1.4	Thesis Overview	14
2	Theoretical Background	16
2.1	Introduction	16
2.2	Vision System Geometry	16
2.2.1	Perspective Projection Image Formation Model	17
2.2.2	Reference Frames and Coordinate Transformations	19
2.2.3	Representations of Relative Orientation	22
2.3	Kalman Filtering Review	30
2.3.1	Discrete-Time Kalman Filter	31
2.3.2	Discrete-Time Extended Kalman Filter	35
2.3.3	Continuous-Discrete Extended Kalman Filter	37
2.3.4	Local Iterations	40
2.4	Cramer-Rao Covariance Bounds	41
2.5	Chapter Summary	47

3	Planar Motion Viewed From The Plane	48
3.1	Introduction	48
3.2	Broida and Chellappa's Approach	54
3.2.1	System Geometry	54
3.2.2	Motion and Structural Model	56
3.2.3	Measurement Model	57
3.2.4	Results and Performance With Simulated Imagery	58
3.2.5	Discussion	74
3.3	Motion Estimation Under Known Structure	75
3.3.1	System Geometry	76
3.3.2	Motion Model	76
3.3.3	Measurement Model	78
3.3.4	Feature Point Occlusion	79
3.3.5	Results and Performance With Simulated Imagery	80
3.3.6	Discussion	92
3.4	Chapter Summary	93
4	Planar Motion, Arbitrary Viewing Angles	94
4.1	Introduction	94
4.2	Broida and Chellappa's Approach	100
4.2.1	System Geometry	101
4.2.2	Motion and Structure Model	101
4.2.3	Measurement Model	107
4.2.4	Results and Performance With Simulated Imagery	110
4.2.5	Discussion	133
4.3	Motion On A Known Planar Surface	134
4.3.1	System Geometry	135
4.3.2	Motion and Structure Model	135

4.3.3	Measurement Model	140
4.3.4	Structure Estimation With Feature Point Occlusion	143
4.3.5	Filter Initialization	145
4.3.6	Results and Performance With Simulated Imagery	146
4.3.7	Discussion	172
4.4	Chapter Summary	176
5	Conclusions and Proposals For Future Work	178
	Reference .	187
A	Simulation Software	194
A.1	Broida and Chellappa's Technique	197
A.2	Imagery Generation	209
A.3	Proposed Planar Motion Technique	233

List of Tables

3.1	Simulation parameters for Case 3.2-1	61
3.2	Simulation parameters for Case 3.2-3	70
3.3	Simulation parameters for Case 3.3-1	82
3.4	Simulation parameters for Case 3.3-2	86
4.1	Simulation parameters for Case 4.2-1	114
4.2	Simulation parameters for Case 4.2-2	125
4.3	Simulation parameters for Case 4.3-1.	149
4.4	Sample initialization statistics for Case 4.3-1.	150
4.5	Simulation parameters for Case 4.3-2.	156
4.6	Simulation parameters for Case 4.3-3.	163

List of Figures

2.1	Central projection image formation geometry.	18
2.2	Three reference frames for motion analysis.	20
3.1	Planar motion viewed from the plane.	49
3.2	Two interpretations of rotational 2D motion.	52
3.3	System geometry for Broida and Chellappa's 2D approach.	55
3.4	Image plane positions of feature points for Case 3.2-1.	63
3.5	Sample results for Case 3.2-1.	63
3.6	Monte Carlo simulation results for Case 3.2-1.	64
3.7	Cramer-Rao bounds for a very wide trajectory.	68
3.8	Cramer-Rao bounds for a very deep trajectory.	68
3.9	Cramer-Rao bound variation with angular velocity.	69
3.10	Image plane positions of feature points for Case 3.2-3.	72
3.11	Sample results for Case 3.2-3.	72
3.12	Monte Carlo simulation results for Case 3.2-3.	73
3.13	System geometry for motion parameter estimation.	77
3.14	Image plane positions of feature points for Case 3.3-1.	84
3.15	Sample results for Case 3.3-1.	84
3.16	Monte Carlo simulation results for Case 3.3-1.	85
3.17	Image plane positions of feature points with occlusion for Case 3.3-2.	88
3.18	Sample results for Case 3.3-2.	88

3.19	Monte Carlo simulation results for Case 3.3-2.	89
3.20	Cramer-Rao bound variation with number of feature points and occlusion.	91
4.1	Two interpretations of rotational 3D motion without occlusion.	97
4.2	Rotational 3D motion with occlusion.	98
4.3	System geometry for Broida and Chellappa's approach.	102
4.4	Object with feature labels used in simulation studies.	111
4.5	True and noisy image sequences for Case 4.2-1.	117
4.6	Sample results for Case 4.2-1.	118
4.7	Monte Carlo simulation results for position and velocity of Case 4.2-1.	119
4.8	Monte Carlo simulation results for orientation and angular velocity of Case 4.2-1.	120
4.9	Monte Carlo simulation results for structure of points p^3 and p^7 of Case 4.2-1.	121
4.10	Monte Carlo simulation results for structure of points p^1 and p^5 of Case 4.2-1.	122
4.11	True and noisy image sequences for Case 4.2-2.	127
4.12	Sample results for Case 4.2-2.	128
4.13	Monte Carlo simulation results for position and velocity of Case 4.2-2.	129
4.14	Monte Carlo simulation results for orientation and angular velocity of Case 4.2-2.	130
4.15	Monte Carlo simulation results for structure of points p^3 and p^7 of Case 4.2-2.	131
4.16	Monte Carlo simulation results for structure of points p^1 and p^5 of Case 4.2-2.	132
4.17	System geometry for planar motion analysis.	136
4.18	True image sequence and occlusion map for Cases 4.3-1 and 2.	152
4.19	Cramer Rao bound variation with number of feature points.	153

4.20	Sample results for Case 4.3-1 with single frame initialization.	154
4.21	Sample results for Case 4.3-1 with all zero initialization.	154
4.22	Monte Carlo simulation results for Case 4.3-1.	155
4.23	Sample results for Case 4.3-2.	159
4.24	Motion parameter Monte Carlo simulation results for Case 4.3-2. . . .	160
4.25	Structure parameter Monte Carlo simulation results for feature points p^3 and p^7 for Case 4.3-2.	161
4.26	Structure parameter Monte Carlo simulation results for feature points p^1 and p^5 for Case 4.3-2.	162
4.27	True image sequence and sample noisy image for Case 4.3-3.	166
4.28	Occlusion map for trajectory of Case 4.3-3.	167
4.29	Sample results for Case 4.3-3.	168
4.30	Motion parameter Monte Carlo simulation results for Case 4.3-3. . . .	169
4.31	Structure parameter Monte Carlo simulation results for feature points p^3 and p^7 for Case 4.3-3.	170
4.32	Structure parameter Monte Carlo simulation results for feature points p^1 and p^5 for Case 4.3-3.	171
4.33	Comparison of rotation operator convergence index RMSE for Case 4.2- 2 and Case 4.3-2.	175
4.34	Comparison of rotation operator convergence index RMSE for Case 4.3- 3.	175

Chapter 1

Introduction

A primary function of biological vision systems is the detection, processing, and understanding of motion perceived from a continuous flow of visual information. Perceived motion results from relative motion of a vision system within a dynamic environment. Analysis of object motion in scenes, or the apparent motion perceived by a moving observer in a static scene, can yield vital information concerning the structure and dynamic behavior of the observed process. Over the last two decades, research in motion detection, analysis, and understanding techniques with applications in computer vision systems for robotics and autonomous vehicles represents an area of vigorous growth. Indeed, during the period 1986 through 1989 inclusive, Rosenfeld's bibliographical surveys (see [1] for example) have cited in excess of four hundred publications on motion analysis in time-varying imagery.

Early work in image sequence analysis prior to 1978 has been surveyed by Martin and Aggarwal [2] and Nagel [3]. A large amount of research during this period focussed on applications of motion analysis to bandwidth compression for transmission of TV signals, and on detection and tracking of geographical landmarks and cloud motion in satellite imagery. In a more limited sense, real-time feature tracking techniques were also studied in applications of visual feedback to industrial automation and autonomous vehicle navigation. Feature tracking methods initially focussed on

extracting motion parameters through detection of object displacement between two consecutive frames—a technique that is still being investigated today. Some emphasis, however, shifted towards tracking features within a sequence of more than two image frames through stochastic modeling of the dynamic processes being observed. Sequential estimation theory, or Kalman filtering, could then be employed to put the tracking process on a firm statistical basis.

This thesis investigates applications of Kalman filtering techniques to the estimation of motion and structural parameters of objects seen in monocular (single camera) image sequences. Object motion parameters include position, orientation, and time derivatives of position and orientation of an object with respect to a specified reference frame. Object structure, in the present context, refers to the shape of the observed object in the sense that positions of specific features are estimated with respect to an object-centred reference frame. These notions are made more precise in Chapter 2. A primary contribution of this thesis is the development of motion, structure, and measurement models, and a corresponding recursive nonlinear estimation procedure appropriate to the case of object motion which is constrained to lie on a known planar surface. The significance of image-based planar motion analysis is apparent from the broad range of robotics applications in which objects of interest are constrained, at least locally in space and time, to lie on a planar surface. Representative examples include the case of objects moving along a conveyor belt or assembly line, aerial surveillance of ground-based objects, and mobile robotics or autonomous vehicle applications in which the robot or vehicle moves on the floor or (locally) flat ground. In the last two examples, solutions must be formulated in the presence of a moving observer (egomotion). The motion and structure estimation technique developed in this thesis can immediately be extended to the case of observer and object motion.

This introductory chapter provides a general review of motion analysis techniques through a review of the literature. Section 1.1 describes and compares two fundamentally different approaches to motion analysis known as image flow-based and feature-based techniques. Section 1.2 presents a summary of major contributions to motion analysis which are based on extended Kalman filtering. This review leads

to the thesis statement in Section 1.3 which identifies several important problems in machine vision applications that have not been adequately studied in the literature. Section 1.4 summarizes this introductory chapter and provides an overview of the remainder of the thesis.

1.1 Two Fundamental Approaches to Motion Analysis

Motion analysis techniques are often classified according to whether they are formulated through an image flow-based, or feature-based approach. The image flow approach first estimates apparent velocities of intensity patterns in an image sequence before motion analysis begins. In feature-based algorithms, features such as points, lines, contours, corners, etc., are extracted from each image and matched between images in order to estimate relative motion of features in scenes with respect to an observer. More recent surveys of methods in motion analysis have been presented by Nagel [4], who focuses on techniques based on measurement of image flow, and Aggarwal [5], who treats primarily feature-based techniques. Feature matching between successive images is known as the *correspondence problem* and has been reviewed by Aggarwal, Davis, and Martin [6].

Image flow is the velocity field in the image plane that arises due to projections of moving patterns in scenes onto the image plane. Flow-based techniques for motion estimation focus first on estimating this velocity field. Image flow is often modeled by the image flow constraint equation [8] which describes the interaction between velocity fields and spatial and temporal gradients of image intensity functions under a common assumption of small motion between successive image frames. This approach requires that spatially and temporally discrete scenes be differentiated through approximate differencing techniques which tends to enhance noise present in real images. The image velocity field is then iteratively computed for each pixel based on various proposed optimization criteria [8, 9, 13]. Further difficulties are encountered in the accurate measurement of flow fields near discontinuities in image

intensity and in areas of uniform intensity [8]. Moreover, if the inter-frame motion is indeed small, inaccuracies in flow field estimates may overshadow the true velocity field leading to ambiguous results.

The image flow field contains information concerning relative depth, surface orientation, and relative motion of objects with respect to an observer [10]. Several methods have been proposed to extract three dimensional structure and motion from flow fields [11, 12, 14]. It has been noted, however, that the system of equations relating position and motion in space to image position and flow are not well behaved numerically and are highly sensitive to noise in image position and flow estimates [5].

Feature-based approaches to motion estimation formulate systems of equations relating positions of features detected in an image sequence to object translational and rotational parameters. Roach and Aggarwall [15] give a thorough discussion of this problem when only two or three frames are considered. Motion parameters are estimated with a modified least-squared error solution of a nonlinear system of equations. Their results demonstrate that this approach can be quite sensitive to noise in estimates of feature positions in the image. Tsai and Huang [16, 17] have shown that in many cases motion parameters can be obtained by solving a system of linear equations for a set of "essential parameters", followed by a Singular Value Decomposition of the 3×3 essential parameter matrix. Their simulation results also show that motion parameter estimates can be highly sensitive to feature position errors. Fang and Huang [18] have implemented a similar approach, which iteratively solves a nonlinear system of equations for motion parameters, and give detailed experimental results. Again, accuracy of the results is seriously degraded in the presence of feature position measurement errors.

A common assumption [5] [15]-[17] [24]-[33] in feature-based techniques is that feature detection and correspondence problems have already been solved. In this sense, detection and correspondence problems are often treated separately from structure and motion estimation. This fact is in sharp contrast to the belief that structure and motion processes, and the correspondence process are complementary, perhaps even inseparable, in the human visual system [50]. Feature detection and

correspondence are required to interpret raw image data and drive the motion estimation scheme. Knowledge of motion parameters and their uncertainty, on the other hand, may serve as constraints to reduce the search space associated with detection and correspondence problems. Matthies and Kanade [35] give a thorough discussion of this concept and provide two illustrative examples which propose solutions based on Kalman filtering techniques. The Kalman filter with its recursive structure and explicit modeling of uncertainty has been proposed as an extremely useful approach to dynamic vision problems. The following section describes major contributions in this area.

1.2 Review of Motion Analysis Through Extended Kalman Filtering

One of the earliest applications of Kalman filtering to image analysis was reported by Futrelle and Speckert [19]. Their work focussed on tracking edges in images through Kalman filtering. The state and covariance extrapolation stage of the filter was used to predict positions of edges in the next image frame in order to reduce the complexity of detection and correspondence problems. However, the treatment in [19] is mostly qualitative and no results are shown.

Gennery [20] conducted studies of tracking the position and orientation of a three-dimensional (3D) known object through measurements taken in stereo image sequences. A recursive algorithm was employed, similar to a Kalman filter, to estimate the position, orientation, velocity, and angular velocity of a known object. Feature locations in future image frames were predicted by the estimation process thereby simplifying the detection and correspondence problems. Measurement and prediction uncertainty was combined to weight most recent measurements and update the state estimate.

Hallam [21] estimated motion parameters for a continuously moving observer by tracking stationary objects in the environment. Relative observer-to-target positions and velocities were measured by a sonar system. Each available target was tracked

independently by Kalman filters, the outputs of which were passed to a central Kalman tracking filter which estimated the observer position and orientation relative to a stationary global reference frame. The number of targets seen in a given sonar scan is variable and, because of specularity and relative motion of observer and targets, a given target will be seen only intermittently. As a result, there is an implicit segmentation problem in matching target observation events to the correct tracking filter. Again, the prediction capability of the Kalman filter was used to solve this correspondence problem. The treatment in [21] is mostly by qualitative discussion; the assumed system and measurement models have not been presented. Some tabulated results are given which seem to indicate satisfactory performance.

Legters and Young [22] have presented a mathematical image model and develop, through an operator formulation, a representation for 2D rigid motion parallel to the image plane. A primary assumption is that a mask is available to convert gray level intensities to binary images. Difference scenes formed from successive binary images are expressed in terms of translational and rotational operators assuming small motion between consecutive scenes. A least-square error criterion is minimized with a variational algorithm to produce an optimal estimate of the unknown parameters of the translation and rotation operators. They have noted that if the moving object is occluded by the foreground, parameter values produced by the variational algorithm become inaccurate. Independent Kalman filters, one for translation and another for rotation, are employed to smooth and stabilize the estimation of tracking parameters during occlusion. Initial estimates for their variational algorithms are taken from the Kalman prediction. Least-squared errors resulting from the variational algorithms provide an estimate of measurement uncertainty for use in the measurement update stage of the filters. Their algorithm has been applied to synthetic binary images and showed encouraging results. However, the variational algorithms operate primarily on leading and trailing edges of difference scenes and, therefore, may be largely dependent on the accuracy with which the binary image sequence is formed from full gray level images. Moreover, due to a strict dependence on difference scenes, algorithm extension to the case of a moving observer is not immediate.

Stuller and Krishnamurthy [23] have applied Kalman filtering to tracking an

object in 2D television images. Localized object intensity regions are modeled as planes whose parameters are computed by least-squares techniques. The availability of these parameters enabled linear measurement models to be formulated and had the added effect of reducing filter sensitivity to image noise. They utilize a non-uniform image motion model of a moving object to estimate translational displacement in consecutive image frames. Their technique provides an estimate of localized image flow in the vicinity of a moving object.

Wu *et al.* [32] have investigated the problem of locating a rigid object and estimating its motion in 3D. They assume that structure is known in the sense that feature point positions on the object are known relative to an object-centred reference frame. They also assume that detection and correspondence problems have been solved. Nonlinear dynamical equations are developed which describe the 3D position and Euler angle orientation of this object-centred reference frame. An artificial measurement sequence based on measurements of feature positions in each image is employed to help reduce the effects of the nonlinear perspective projection image formation model on filter performance. An extended Kalman filter is developed to estimate the object's position, translational velocity, orientation, and angular velocity. Their technique has been applied to real scenes generated in a laboratory environment and showed encouraging results. It is interesting to note that these results were generated by using 14 feature points on an object of known structure which is constrained to lie on a planar surface. In Chapter 4 of this thesis, it is demonstrated that very little additional information is provided to motion parameter estimation as the number of feature points is increased beyond 4.

Chang *et al.* [33] investigate the problem of tracking a point target from measurements obtained with correlated noise processes. They estimate state variables (position, velocity, and acceleration) in 3D of a manoeuvring target trajectory which is disturbed by an unknown acceleration. A first-order difference operator is applied to the correlated measurement sequence to produce an artificial measurement sequence with only uncorrelated white noises. An extended Kalman filter augmented by a semi-Markov process and Bayesian theory is applied to form an adaptive filter. The artificial measurement sequence up to the current time is used to predict the

a posteriori probability of each possible acceleration state. These probabilities are used to form a weighted average of acceleration and derive optimal estimates of state parameters. Simulation results are shown which demonstrate the effectiveness of these modifications to the standard Kalman filtering approach.

Ayache and Faugeras [34] recursively build and update 3D representations of the environment of a mobile robot. Although methods in [34] are developed for trinocular (three cameras) image sequences, their approach could also be applied to stereo imagery or that produced by a monocular imaging system under egomotion. Positions of world features (points, lines, planes) are recursively estimated from measurements in the image plane through extended Kalman filtering. They have generalized the problem of updating positions of any feature so that the same Kalman filtering structure can be employed in all cases. Geometric relations are inferred between features by computing a generalized Mahalanobis distance and using a χ^2 acceptance test. An estimate of robot motion is also maintained with the Kalman filter. They have applied their technique to real scenes taken from an indoor mobile platform and demonstrate some of the major properties of their algorithm.

Matthies, Kanade, and Szeliski [36] have recently applied Kalman filtering to the problem of recursively building dense depth maps from image flow measurements for mobile robotics applications. Assuming only lateral motion and small interframe displacements, a two dimensional flow vector is computed at each image point. A correlation-based algorithm followed by cubic interpolation along scan lines is then used to establish flow estimates to sub-pixel accuracy. They note that, especially in areas of uniform intensity, the raw depth or disparity (inverse depth) values obtained from flow measurements can be very noisy. In order to reduce the effects of this noise, they smooth the disparity map with piecewise continuous splines. The smoothing algorithm is implemented with discontinuity detection to reduce blurring of the depth map across object boundaries. A scalar Kalman filter for each pixel element provides the recursive mechanism to estimate depth and depth uncertainty maps. They have demonstrated the effectiveness of their technique on real image sequences. However, the significant computational and data storage requirements of this approach restrict its practical applications to highly parallel

computing architectures.

Dickmanns and Graefe [37] present a thorough discussion of their approach to dynamic machine vision which employs special hardware and methods for feature extraction and information processing. Through the use of integral spatio-temporal models, extended Kalman filtering, and prediction error feedback, they have demonstrated real-time vision-based control capabilities in several important problems in robotics [38]. Real-time capability in dynamic vision is achieved through a specialized multiple instruction stream, multiple data stream (MIMD) computer architecture [38, 39]. This system associates each processing element with a dynamically allocated window on each image which, in turn, is associated with a single feature of interest. Each window is defined by prediction error feedback from the Kalman filter. Controlled correlation techniques are employed by each processing element to efficiently extract the feature of interest in each window. Results of feature extraction are passed from all processing elements to a host computer which executes the extended Kalman filter. They have successfully applied their technique to control of an inverted pendulum on a moving cart [40], autonomous vehicle docking problems [38], autonomous road vehicle guidance [38], and vision-based control of an aircraft in a simulated landing approach [41]. More recent extensions of algorithms and hardware to obstacle avoidance in autonomous vehicle guidance applications are described in [42]. This research activity also exploits planar motion constraints in the formulation of motion and measurement models with monocular imaging systems.

Broida [31], Broida and Chellappa [24, 25, 26, 27], Broida, Chandrashekhar, and Chellappa [28], and Young and Chellappa [29, 30] have applied extended Kalman filtering techniques to the estimation of motion and structural parameters of a rigid body through measurements taken from noisy monocular and stereo image sequences. In [24], a special case of the general motion problem is considered in which object translational motion is constrained to lie on a plane defined by the optical axis and central scan line of the imaging system. All feature points of interest are assumed to lie on this plane. Through dynamic modeling of the moving object, estimates of model parameters are refined over an arbitrary number of image frames with an iterated extended Kalman filter under the assumption that some

prior knowledge of object structure is available. Assuming that feature detection and correspondence problems have been solved, they present encouraging simulation results. This technique is reviewed in Chapter 3, Section 3.2, and simulation results are shown. In [25]-[28], system and measurement models are derived for more general motion. Quaternions are employed to model rotational motion under a claim that an earlier choice of Euler angles and rates was neither simple nor well-behaved. In [25], simulation results are shown for purely translational motion in which a decoupled filter was employed based on prior knowledge that motion was translational only. They also treat the special case previously considered in [24]. Further investigation of this technique is presented in [28] where constant translational and rotational velocities are assumed. The methods of [25, 28] are reviewed in Chapter 4, Section 4.2, and simulation results are shown. In contrast to the recursive algorithms presented in [24, 25, 28, 29, 30], a batch approach based on a fixed number of frames is developed in [26]. Approximate Cramer-Rao covariance bounds in the absence of *a priori* information are developed and analyzed in [27]. Young and Chellappa have extended this technique in [29] to stereo imaging systems and treat in more detail the development of motion and measurement models, the estimation algorithm, and uniqueness issues in [30].

Application of sequential state estimation techniques such as the Kalman filter to machine vision problems emphasizes temporal continuity conditions for image sequence interpretation. This allows proper definitions of state variables and formulation of systems of differential equations to model motion processes of objects in space and time. Dickmanns [37] refers to spatio-temporal motion analysis as "dynamic vision" which, in contrast to "static" image sequence processing, does not separately apply object recognition from one frame to the next as a first step and motion reconstruction afterwards as a second. Instead, object structure and motion are treated simultaneously in order to generate an estimate of the system state based on measurements from noisy images. There are several immediate advantages of this approach:

1. Storage requirements for past images are reduced or eliminated in that the current system state can be represented solely by state vectors and their un-

certainty, together with shape descriptors which describe the particular feature to which each state vector applies (point, line, plane, etc.);

2. The motion estimation process is not explicitly dependent on assumptions of small motion between scenes which is common to the optical flow based approach [8, 9, 11, 12, 13];
3. The estimation process can be made robust to large measurement errors through explicit modeling of uncertainty in measurement and dynamic models; and
4. Feedback of state predictions and uncertainties reduces the search space associated with feature detection and correspondence problems.

Two disadvantages of this technique arise from computational requirements of Kalman filters, and nonlinear measurement models which result in the use of suboptimal extended Kalman filters. However, the computational burden may, in some cases, be less than that required for iterative approaches to obtain solutions to systems of nonlinear equations. A performance evaluation of these suboptimal estimation techniques is one of the primary objectives of this thesis.

The methods reviewed above have been developed with somewhat different applications in mind. Each method is dependent on a particular set of conditions which is assumed to be known *a priori*. For example, different types of imaging sensors have been investigated: sonar [21], visual monocular [19] [22]-[28] [32, 33, 36, 37], or visual stereo and trinocular [20, 29, 30, 34, 37]. Some approaches assume a static environment with moving sensor [21, 34, 36], while others consider dynamic environments with stationary sensors [19, 20, 22] [24]-[33], and some consider both a dynamic environment and moving sensor [23, 37]. In some cases only motion parallel to the image plane is considered [22, 23, 36]. Of those techniques which estimate motion and orientation of objects in scenes, some assume partial or complete knowledge of object structure [19, 20, 24, 32] while others estimate total object structure from stereo images or with monocular image sequences assuming depth information is available [28, 29, 30]. A general result is that performance of the above techniques cannot be directly compared since each must be applied to the specific problem for which

it was developed. However, the research efforts of Broida and Chellappa [24, 28], Young and Chellappa [30], Ayache and Faugeras [34], and Dickmanns and Graefe [37, 42] may readily serve as foundations on which to base further developments of dynamic vision systems for a wide range of robotics applications.

Evaluation of proposed estimation schemes using real imagery has been conducted extensively only in the work of Ayache and Faugeras [34], Mathies *et al.* [36], and Dickmans *et al.* [38]. In a more limited sense, Wu *et al.* [32] show results for imagery obtained in a controlled laboratory setting, while Broida [31] and Broida *et al.* [28] show qualitative results based on real imagery for which ground truth data is not available, and feature points were extracted manually. In most of the above investigations, simulations are performed in which feature detection and correspondence problems are assumed to be solved. In this case, feature positions in each image are obtained from true analytical motion and structural models and an assumed image formation model, and additive noise processes are included to simulate measurement noise.

1.3 Thesis Statement

The literature review of the previous section has noted a wide range of assumptions concerning the nature of assumed prior information. For example, in Broida and Chellappa's approach [24, 28], object motion is assumed to be unforced with constant translational and rotational velocities. In a large number of robotics applications the objects of interest are constrained to lie on some spatial surface which is known or can be approximated by analytical expressions. This may often be the case in aerial surveillance of ground-based objects, in industrial robotics where objects move on conveyor belts or along assembly lines, and in most mobile robotics applications where the robot itself must travel on the floor or ground. Moreover, it is often the case that the surface on which motion takes place is planar, or at least can be approximated as such locally over space and time. This problem is quite distinct from those considered in [22, 23, 36] in which motion takes place in a plane parallel to the image plane of the sensor. It is somewhat surprising, given the broad range

of applications to which planar constrained motion would apply, that an explicit analysis of this problem is not present in the literature.

Another problem of fundamental importance in feature-based techniques is that of feature occlusion due to object motion or other objects in the foreground of a scene. A common approach to the occlusion problem is to assume that it does not occur [7]. Broida and Chellappa [24, 28, 31], on the other hand, recognize the fact that a Kalman filtering approach may provide natural solutions in the presence of occlusion, but their analysis proceeds under a transparent object assumption so that occlusion need not be considered. The effect of feature occlusion on the estimation process, therefore, has not been examined.

In robotics applications, it is often the case that the camera itself is in motion. An equivalent problem results from applications in which it is desirable to estimate object motion with respect to a reference system attached to the robot. In fact, dynamic models for constrained object motion can often be simplified if motion parameters are expressed with respect to a reference system distinct from that associated with the camera. Proposed solutions in this context are not generally available in the literature.

This thesis investigates the application of extended Kalman filtering to object motion and structure estimation through measurements of feature positions in noisy monocular image sequences. Specifically, this analysis has four primary objectives:

1. To review developments of methods proposed by Broida and Chellappa in [24] and [28], and to evaluate these techniques through simulation studies;
2. To propose an alternate parameterization for structure and motion processes specific to the case of motion on a known plane, and to demonstrate through simulation studies the performance that might be expected with this new approach in comparison to the more general methods in [28];
3. To study the effects of feature occlusion, rotational rates, and the number of available feature points on the proposed motion and structure estimation technique; and

4. To formulate for the motion estimation problem a solution which is immediately extended to the case of a moving observer.

A complete evaluation of either Broida and Chellappa's method or the proposed technique on real imagery is beyond the scope of the thesis work. Such an analysis will require an intense review of feature detection and correspondence problems, sensor modeling and measurement noise characterization, and complex experimental arrangements in order to obtain long sequences (100 frames, for example) of digital, time-aligned, recorded imagery and ground truth data associated with object and/or camera motion. More complete implementations and performance analysis with real imagery in robotics and aerial surveillance applications constitute major proposals for future research. In this thesis, suitable imagery with or without occlusion is computer-generated for a translating and rotating block. Features of interest are the block corners whose positions in each image are perturbed by an additive pseudo-random noise process and made available to structure and motion estimation algorithms along with correspondence information. This approach to simulation analysis is consistent with similar studies in the literature [15] [24]-[30].

1.4 Thesis Overview

This chapter introduced and compared image flow and feature based approaches to motion analysis, and presented a survey of major contributions which apply extended Kalman filtering to motion and structure estimation. Three problems which have not been adequately studied in the literature, but which are of widespread interest in machine vision applications have been identified: motion and structure estimation under a planar motion constraint, effects of feature occlusion on the estimation process, and solutions in the presence of egomotion.

The remainder of this thesis is organized as follows:

Chapter 2 first introduces the assumed geometry and notation used for reference frames and coordinate transformations, and provides a review of common parameterizations for change-of-basis transformations, or rotation matrices.

This is followed by a review of linear and extended, discrete and continuous Kalman filters, and local iterations used in the iterated extended Kalman filter employed in [24, 28]. Error covariance lower bounds based on the Cramer-Rao inequality are derived for the nonlinear, discrete-time estimation process. These bounds provide a performance measure for filters derived and evaluated in Chapters 3 and 4.

Chapter 3 considers a special case of the planar motion problem with the specific camera viewing angles employed in [24]. Broida and Chellappa's method in [24] is reviewed and simulation results are shown. An alternate parameterization of object motion is proposed which fully exploits the rigid object assumption. Estimation of object structure is removed and emphasis is placed on motion parameter estimation with and without occlusion for the proposed motion model.

Chapter 4 investigates constant velocity motion in the plane with arbitrary viewing angles. Broida and Chellappa's method of [28] is reviewed and simulation results are shown. Linear motion and structural models, and a nonlinear measurement model are proposed for the case in which object motion is constrained to a known plane. Simulation results for this technique are presented for both known and unknown structure, and results are compared to those obtained for Broida and Chellappa's method.

Chapter 5 provides general discussions, proposals for future research, and conclusions of the thesis work.

Chapter 2

Theoretical Background

2.1 Introduction

This chapter reviews fundamental theory required for development of solutions to motion analysis problems considered in subsequent chapters. Section 2.2 introduces the assumed geometry and notation, and provides a review of common parameterizations for change-of-basis transformations. It should be noted that the axis orientation for the vision system, defined below in Section 2.2.1, has been selected primarily for use with aerial surveillance systems and, therefore, differs from that which is commonly used in the literature. Section 2.3 provides a review and discussion of the linear discrete-time Kalman filter, the discrete-time extended Kalman filter, and the continuous-discrete Kalman filter. Local iterations which lead to the iterated extended Kalman filter employed in [24, 28] are also introduced. Section 2.4 derives approximate error covariance lower bounds based on the Cramer-Rao inequality for the nonlinear, discrete-time estimation process. These bounds provide a performance measure for filters derived and evaluated in Chapters 3 and 4.

2.2 Vision System Geometry

This section introduces the perspective projection image formation model, and notation for reference frames and coordinate transformations. The nine parameter, Euler roll-pitch-yaw, exponential, and quaternion representations for object orien-

tation and rotational motion are discussed in this context.

2.2.1 Perspective Projection Image Formation Model

If optical distortions of the imaging device are not severe, a perspective projection image formation model can be used to relate the observed location of a feature in a 2D image to the 3D coordinates of that feature in a camera-centred reference frame. Figure 2.1 shows the assumed geometry for the image formation model. The position of the i th feature point, p^i , with respect to a camera-centred reference frame, F_C , is described by the vector r_C^i . A feature at point p^i is projected onto the image plane and appears at point (L, u^i, v^i) in F_C , where L is the focal length. The origin of F_C is taken at the centre of projection of the imaging lens. The x_C axis of F_C is aligned with the optical axis, while y_C is parallel to a scan line and positive to the right in the image. The z_C axis is positive down in the image to form a right-handed coordinate system.

In applications, measurements of feature locations are taken from a spatially discrete image stored in computer memory. In this case, horizontal and vertical scale factors and other distortion effects which define a mapping from the imaging sensor array to this stored representation must also be considered. For the purposes of simulation, distortion effects are often assumed to be negligible while all scale factors are set to unity. Under these assumptions, and with

$$r_C^i = \begin{bmatrix} r_{C,1}^i \\ r_{C,2}^i \\ r_{C,3}^i \end{bmatrix}, \quad (2.1)$$

the image plane position of the projection of the point p^i is given by

$$\begin{aligned} u^i &= L \left(\frac{r_{C,2}^i}{r_{C,1}^i} \right), \text{ and} \\ v^i &= L \left(\frac{r_{C,3}^i}{r_{C,1}^i} \right). \end{aligned} \quad (2.2)$$

The central projection transformation defined in (2.2) demonstrates the loss of depth information in monocular imaging systems; any scalar multiple of r_C^i will result in

Figure 2.1: Central projection image formation geometry. A feature at point p^i projects onto the point (L, u^i, v^i) in the camera-centred frame, F_C , where L is the focal length.

the same image position of the feature at point p^i . As a result, position and time derivatives of position, and object structure can only be recovered up to a scale factor in the absence of prior information.

It is common practice in simulation analysis to set the focal length L to unity without loss of generality. However, in some applications, imaging systems with variable focal length are required. For clarity, the focal length is retained throughout this analysis.

2.2.2 Reference Frames and Coordinate Transformations

Under a rigid object assumption it is convenient to represent feature points on the object as constant vectors in an object-centered reference frame. Object "structure" then refers to positions of feature points in an object-centred frame. The image formation model described in Section 2.2.1 requires that feature points be expressed in terms of a camera-centred reference frame. A common approach is to express object position, orientation, and motion parameters with respect to the camera frame, in which case only relative motion is of interest. However, in a large number of applications motion analysis is required in the presence of egomotion. Moreover, it is often preferred in robotic applications that object position and motion be described with respect to a reference system on the robot. In this case, it is desirable to express both camera position and motion as well as object motion with respect to an earth-fixed, or stationary reference frame. Figure 2.2 shows the three coordinate systems that will be used throughout this analysis. These three reference frames are denoted by

$$\begin{aligned} F_C &\sim \text{Camera-centred} \\ F_O &\sim \text{Object-centred, and} \\ F_E &\sim \text{Earth-fixed reference frame.} \end{aligned} \tag{2.3}$$

A reference frame, F_α , where in this context α represents either C , O , or E , is defined by a point O_α , the origin of F_α , and a right-handed, ordered, orthonormal basis set $\{i_\alpha, j_\alpha, k_\alpha\}$. The unit vectors i_α , j_α , and k_α lie along the x_α , y_α , and z_α axis, respectively. A vector r , when expressed with respect to F_α , will be denoted by

Figure 2.2: Three reference frames for motion analysis. The three frames F_E , F_O , and F_C denote earth-fixed, object-centred, and camera-centred reference frames, respectively. All frames are right-handed orthonormal triads with origin O , and are related by translation vectors T and orthogonal change-of-basis transformations I .

r_α . The components of r_α are written as

$$r_\alpha = \begin{bmatrix} r_{\alpha,1} \\ r_{\alpha,2} \\ r_{\alpha,3} \end{bmatrix} = r_{\alpha,1}i_\alpha + r_{\alpha,2}j_\alpha + r_{\alpha,3}k_\alpha. \quad (2.4)$$

A second reference frame, F_β , is related to F_α by a translation vector T_β^α and an orthogonal transformation I_β^α . The vector T_β^α represents the translation from O_α to O_β . This translation can be applied only if it is expressed with respect to some reference frame; for example, T_β^α expressed with respect to F_β is written as $[T_\beta^\alpha]_\beta$. The change-of-basis transformation I_β^α can be written as an orthogonal 3×3 matrix which represents a rigid rotation of R^3 . The notation I_β^α denotes a transformation from the α -basis to the β -basis.

Let the position of the i th feature point, p^i , be described in F_O by the vector r_O^i which connects O_O to p^i . This vector can be expressed in terms of the E -basis through the transformation I_E^O , and is written as

$$[r_O^i]_E = I_E^O r_O^i. \quad (2.5)$$

The vector from O_E to p^i expressed in F_E is given by

$$\begin{aligned} r_E^i &= [r_O^i]_E + [T_O^E]_E \\ &= I_E^O r_O^i + [T_O^E]_E. \end{aligned} \quad (2.6)$$

The transformation I_C^E and translation T_C^E can be applied to r_E^i of (2.6) to express the vector from O_C to p^i in F_C as

$$\begin{aligned} r_C^i &= I_C^E \{ I_E^O r_O^i + [T_O^E]_E - [T_C^E]_E \} \\ &= I_C^E \{ I_E^O r_O^i + [T_O^E]_E \} - [T_C^E]_C. \end{aligned} \quad (2.7)$$

The instantaneous position and orientation of the rigid object with respect to the earth-fixed reference frame are given by the translation T_O^E and the transformation I_O^E , respectively. Similarly, the position and orientation of the camera with respect to this earth-fixed frame are defined by T_C^E and I_C^E , respectively, at each instant in

time. It is well-known that $\mathbf{I}_C^E$ can be written as

$$\begin{aligned}\mathbf{I}_C^E &= \begin{bmatrix} [i_E]_C & [j_E]_C & [k_E]_C \end{bmatrix} \\ &= \begin{bmatrix} [i_C]_E & [j_C]_E & [k_C]_E \end{bmatrix}^T, \end{aligned} \quad (2.8)$$

where superscript 'T' denotes transposition. With this notation, the i th feature point in F_C is described by $\mathbf{r}_C^i = [r_{C,1}^i, r_{C,2}^i, r_{C,3}^i]^T$, where

$$\begin{aligned}r_{C,1}^i &= [i_C]_E^T \{ \mathbf{I}_E^O \mathbf{r}_O^i + [\mathbf{T}_O^E]_E - [\mathbf{T}_C^E]_E \}, \\ r_{C,2}^i &= [j_C]_E^T \{ \mathbf{I}_E^O \mathbf{r}_O^i + [\mathbf{T}_O^E]_E - [\mathbf{T}_C^E]_E \}, \text{ and} \\ r_{C,3}^i &= [k_C]_E^T \{ \mathbf{I}_E^O \mathbf{r}_O^i + [\mathbf{T}_O^E]_E - [\mathbf{T}_C^E]_E \}. \end{aligned} \quad (2.9)$$

The representation in (2.9) allows direct substitution into the image formation model defined in (2.2).

In a feature-based approach to motion analysis, the structure of a rigid object can be defined by a set of constant vectors, $\mathcal{S} = \{\mathbf{r}_O^i | i = 1, 2, \dots, N_p\}$, where N_p is the total number of feature points of interest. In general, N_p is a non-decreasing, integer-valued function of time; as new feature points not already in $\mathcal{S}$ are recognized, they can be added to the structural model. At any given time, however, only a subset $\mathcal{S}_v \subseteq \mathcal{S}$ may be visible due to occlusion of feature points.

2.2.3 Representations of Relative Orientation

The previous section defined a change-of-basis transformation $\mathbf{I}_\beta^\alpha$ such that a vector $\mathbf{r}_\alpha$ in F_α is transformed to $[\mathbf{r}_\alpha]_\beta$, its representation in F_β , through the expression

$$[\mathbf{r}_\alpha]_\beta = \mathbf{I}_\beta^\alpha \mathbf{r}_\alpha. \quad (2.10)$$

The orthogonal transformation $\mathbf{I}_\beta^\alpha$ belongs to the three-dimensional rotation group for which there are about eight commonly used representations [48]. Four such representations are reviewed here: the nine-parameter expression; Euler parameter formulations; exponential representations; and the quaternion expression.

Nine Parameter Representation: This representation is simply the nine components of the 3×3 matrix $\mathbf{I}_\beta^\alpha$ which are denoted as

$$\mathbf{I}_\beta^\alpha = [l_{ij}]_\beta^\alpha, \quad (2.11)$$

where l_{ij} is the cosine of the angle from the i th basis vector of F_α to the j th basis vector of F_β . These nine parameters are subject to six quadratic constraints, which arise from the orthogonality of $\mathbf{I}_\beta^\alpha$, leaving three degrees of freedom.

Roll Pitch Yaw Representation: Because a rigid body possesses three rotational degrees of freedom, three independent parameters should be sufficient to completely and unambiguously describe the orientation of the body. The roll-pitch-yaw representation [32, 46, 59] is a particular case of an Euler angle parameterization of the three-dimensional rotation group. If F_β is attached to the rotating body and F_α is the reference system, then the roll, pitch, and yaw angles, denoted ϕ , θ , and ψ , respectively, define an ordered sequence of three plane rotations which can be used to express the α -basis with respect to F_β :

$$\begin{aligned}\mathbf{I}_\beta^\alpha &= \begin{bmatrix} [i_\alpha]_\beta & [j_\alpha]_\beta & [k_\alpha]_\beta \end{bmatrix} \\ &= \mathbf{G}(1, \phi) \mathbf{G}(2, \theta) \mathbf{G}(3, \psi),\end{aligned}\tag{2.12}$$

where

$$\begin{aligned}\mathbf{G}(1, \phi) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & \sin \phi \\ 0 & -\sin \phi & \cos \phi \end{bmatrix}, \\ \mathbf{G}(2, \theta) &= \begin{bmatrix} \cos \theta & 0 & -\sin \theta \\ 0 & 1 & 0 \\ \sin \theta & 0 & \cos \theta \end{bmatrix}, \text{ and} \\ \mathbf{G}(3, \psi) &= \begin{bmatrix} \cos \psi & \sin \psi & 0 \\ -\sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix}.\end{aligned}\tag{2.13}$$

The angles ϕ , θ , and ψ have positive sense defined by the right hand rule about their respective rotation axis. Specifically, the reference system F_α is rotated first about $\mathbf{k}_\alpha$ by the yaw angle ψ , second about the axis $[-\sin \psi, \cos \psi, 0]_\alpha^T$ by the pitch angle θ , and finally about the axis $[\cos \psi \cos \theta, \sin \psi \cos \theta, -\sin \theta]_\alpha^T$ by the roll angle ϕ . In

order to avoid ambiguities in these angles their ranges are often limited to

$$\begin{aligned} -\pi &\leq \psi < \pi, \\ -\frac{\pi}{2} &\leq \theta \leq \frac{\pi}{2}, \text{ and} \\ -\pi &\leq \phi < \pi. \end{aligned} \quad (2.14)$$

A major difficulty with this approach is that the sequence of plane rotations are not defined about orthogonal axis. This property results in highly nonlinear dynamic models which exhibit singularities for particular orientations [46]. The angular velocity of F_β with respect to F_α expressed in F_α , $[\mathbf{w}_\alpha^\beta]_\alpha$, is given in terms of the roll-pitch-yaw rates as

$$[\mathbf{w}_\alpha^\beta]_\alpha = \mathbf{J}_{RPY} \begin{bmatrix} \dot{\psi} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix}, \quad (2.15)$$

where

$$\mathbf{J}_{RPY} = \begin{bmatrix} 0 & -\sin \psi & \cos \psi \cos \theta \\ 0 & \cos \psi & \sin \psi \cos \theta \\ 1 & 0 & -\sin \theta \end{bmatrix}. \quad (2.16)$$

The mathematical singularities, $\theta = \pm\pi/2$, characterize the orientations where the determinant of the Jacobian matrix $\det[\mathbf{J}_{RPY}] = -\cos \theta$ vanishes. When no confusion can result, the angular velocity vector $[\mathbf{w}_\alpha^\beta]_\alpha$ will be written simply as $\mathbf{w}$.

Exponential Representation: A rigid rotation of R^3 is completely defined by a unit vector $\xi = [\xi_1, \xi_2, \xi_3]^T$ which specifies the axis of the rotational operator $\mathbf{I}_\beta^\alpha$, and an angle γ through which the coordinate system is rotated about ξ . This is a four-parameter representation with one quadratic constraint. It is important to note the difference between the axis, ξ , of the rotational operator and what is commonly called the "axis of rotation" or instantaneous angular velocity of the object $\mathbf{w}(t)$. Again, let F_β be attached to the rotating body and F_α be the reference system. If $\mathbf{I}_\beta^\alpha$ rotates R^3 through an angle γ about an axis defined by the unit vector ξ_α to bring F_α into alignment with F_β , then [34, 49]

$$\mathbf{I}_\beta^\alpha = \exp \left(-\gamma \xi_\alpha^\# \right), \quad (2.17)$$

where $\xi_\alpha^{\dagger}$ is the matrix cross product operator associated with ξ_α and is defined by

$$\begin{aligned}\xi_\alpha^{\dagger} &= (\xi_\alpha \times) \\ &= \begin{bmatrix} 0 & -\xi_{\alpha,3} & \xi_{\alpha,2} \\ \xi_{\alpha,3} & 0 & -\xi_{\alpha,1} \\ -\xi_{\alpha,2} & \xi_{\alpha,1} & 0 \end{bmatrix}.\end{aligned}\quad (2.18)$$

By applying the well-known Cayley-Hamilton theorem to the characteristic equation of the skew symmetric matrix $\xi_\alpha^{\dagger}$, and using the fact that ξ_α is a unit vector, the following important identity is obtained:

$$(\xi_\alpha^{\dagger})^3 + \xi_\alpha^{\dagger} = 0. \quad (2.19)$$

A result of (2.19) is that the highest order of any power series of $\xi_\alpha^{\dagger}$ is two. Consequently, (2.17) can be written as

$$\begin{aligned}\mathbf{I}_\beta^\alpha &= \sum_{j=0}^{\infty} \frac{1}{j!} (-\gamma \xi_\alpha^{\dagger})^j \\ &= \mathbf{I}_3 - \sin(\gamma) \xi_\alpha^{\dagger} + [1 - \cos(\gamma)] (\xi_\alpha^{\dagger})^2,\end{aligned}\quad (2.20)$$

where $\mathbf{I}_3$ is the 3×3 identity matrix. This result provides a method to obtain a rotation matrix from its axis and rotation angle. The angle of rotation, γ , can be obtained from $\mathbf{I}_\beta^\alpha$ either by noting that in (2.20) $\xi_\alpha^{\dagger}$ is skew symmetric and¹ $\text{Tr}\{(\xi_\alpha^{\dagger})^2\} = -2$, or by using the fact that eigenvalues of an orthogonal matrix have unity modulus and that the trace of a matrix equals the sum of the eigenvalues. In either case,

$$\cos(\gamma) = \frac{1}{2} (\text{Tr}\{\mathbf{I}_\beta^\alpha\} - 1). \quad (2.21)$$

In addition, the axis ξ_α may be obtained from (2.20) with

$$\xi_\alpha^{\dagger} = \frac{-1}{2 \sin(\gamma)} (\mathbf{I}_\beta^\alpha - (\mathbf{I}_\beta^\alpha)^T). \quad (2.22)$$

Because a rotation about ξ_α through an angle γ is equivalent to a rotation about $-\xi_\alpha$ through an angle $-\gamma$, the angle of rotation may be restricted to the interval $0 \leq \gamma \leq \pi$ for unique inversion of the cosine in (2.21). In this case ξ_α computed from

¹The notation $\text{Tr}\{\mathbf{A}\}$ denotes the trace of the matrix $\mathbf{A}$.

(2.22), together with γ define the transformation I_β^α . However, numerical difficulties may arise when I_β^α is symmetric ($\gamma = 0, \pi$) for which the expression in (2.22) is of indeterminate form (0/0).

Another important identity results from the fact that ξ is a unit vector and hence $\xi^T \xi = 1$. Taking time derivatives of both sides gives $\dot{\xi}^T \xi = \xi^T \dot{\xi} = 0$, which means that ξ is always orthogonal to its time derivative. In this case, for any vector $\mathbf{a}$, $\dot{\xi} \times (\xi \times \mathbf{a})$ is collinear with ξ and hence

$$\xi \times (\dot{\xi} \times (\xi \times \mathbf{a})) = 0. \quad (2.23)$$

In terms of matrix cross product operators, (2.23) is written as

$$\xi^{\dagger} \dot{\xi}^{\dagger} \xi^{\dagger} = 0. \quad (2.24)$$

The identities in (2.19) and (2.24) are required in the derivation of the angular velocity in terms of γ and ξ and their time derivatives.

The angular velocity of F_β with respect to F_α expressed in F_α , $[\mathbf{w}_\alpha^\beta]_\alpha$ can be computed with

$$I_\alpha^\beta = [i_\beta]_\alpha, [j_\beta]_\alpha, [k_\beta]_\alpha, \quad (2.25)$$

and by noting that

$$\begin{aligned} \frac{d[i_\beta]_\alpha}{dt} &= [\mathbf{w}_\alpha^\beta]_\alpha \times [i_\beta]_\alpha, \\ \frac{d[j_\beta]_\alpha}{dt} &= [\mathbf{w}_\alpha^\beta]_\alpha \times [j_\beta]_\alpha, \\ \frac{d[k_\beta]_\alpha}{dt} &= [\mathbf{w}_\alpha^\beta]_\alpha \times [k_\beta]_\alpha, \end{aligned} \quad (2.26)$$

which gives

$$\dot{I}_\alpha^\beta = [\mathbf{w}_\alpha^\beta]_\alpha^\dagger I_\alpha^\beta. \quad (2.27)$$

Rearranging (2.27) gives

$$[\mathbf{w}_\alpha^\beta]_\alpha^\dagger = \dot{I}_\alpha^\beta (I_\alpha^\beta)^T = \dot{I}_\alpha^\beta I_\beta^\alpha. \quad (2.28)$$

Substitution of the transpose of (2.20) into (2.28) followed by repeated applications of identities (2.19) and (2.24) results in

$$[\mathbf{w}_\alpha^\beta]_\alpha^\dagger = \dot{\gamma} \xi_\alpha^\dagger + \sin \gamma \dot{\xi}_\alpha^\dagger + (1 - \cos \gamma) (\xi_\alpha^\dagger \dot{\xi}_\alpha^\dagger - \dot{\xi}_\alpha^\dagger \xi_\alpha^\dagger). \quad (2.29)$$

All terms in (2.29) are cross product operators. Two standard results from vector analysis also give, for any $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbb{R}^3$,

$$\begin{aligned}\mathbf{a} \times (\mathbf{b} \times \mathbf{c}) &= \mathbf{a}^T \mathbf{c} \mathbf{b} - \mathbf{a}^T \mathbf{b} \mathbf{c}, \text{ and} \\ (\mathbf{a} \times \mathbf{b}) \times \mathbf{c} &= (\mathbf{b} \mathbf{a}^T - \mathbf{a} \mathbf{b}^T) \mathbf{c}.\end{aligned}\quad (2.30)$$

With (2.30) and any vector $\mathbf{c} \in \mathbb{R}^3$, and recalling that $\xi^T \dot{\xi} = \dot{\xi}^T \xi = 0$,

$$\begin{aligned}(\xi_\alpha^! \dot{\xi}_\alpha^! - \dot{\xi}_\alpha^! \xi_\alpha^!) \mathbf{c} &= \xi_\alpha \times (\dot{\xi}_\alpha \times \mathbf{c}) - \dot{\xi}_\alpha \times (\xi_\alpha \times \mathbf{c}) \\ &= (\dot{\xi}_\alpha \xi_\alpha^T - \xi_\alpha \dot{\xi}_\alpha^T) \mathbf{c} \\ &= (\xi_\alpha \times \dot{\xi}_\alpha) \times \mathbf{c},\end{aligned}\quad (2.31)$$

which implies that

$$(\xi_\alpha^! \dot{\xi}_\alpha^! - \dot{\xi}_\alpha^! \xi_\alpha^!) = (\xi_\alpha^! \dot{\xi}_\alpha)^{\#}. \quad (2.32)$$

This result allows the cross product operators of (2.29) to be removed which gives

$$[\mathbf{w}_\alpha^\beta]_\alpha = \dot{\gamma} \xi_\alpha + \sin \gamma \dot{\xi}_\alpha + (1 - \cos \gamma) \xi_\alpha^! \dot{\xi}_\alpha. \quad (2.33)$$

Equation (2.33) demonstrates that the angular velocity has components on three orthogonal axis ξ_α , $\dot{\xi}_\alpha$, and $\xi_\alpha^! \dot{\xi}_\alpha$. If the rotational axis is fixed, (2.33) reduces to the simple expression $[\mathbf{w}_\alpha^\beta]_\alpha = \dot{\gamma} \xi_\alpha$.

Another interesting observation is that the Euler angle formulation of (2.12) and (2.13) can be expressed in terms of matrix exponentials as

$$\mathbf{I}_\beta^\alpha = \exp \left\{ -\phi \left(\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \times \right) \right\} \exp \left\{ -\theta \left(\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \times \right) \right\} \exp \left\{ -\psi \left(\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times \right) \right\}. \quad (2.34)$$

This representation for the sequence of plane rotations in (2.12) is particularly convenient when differentiation of $\mathbf{I}_\beta^\alpha$ with respect to roll, pitch, or yaw angles is required, such as in the extended Kalman filtering formulation.

Quaternion Representation: A quaternion [25]-[31] [47]-[49] is a quadruple of ordered² real numbers $\mathbf{q} = [q_1, q_2, q_3, q_4]^T$ associated, respectively, with four units:

²Some authors denote the scalar quaternion element as q_1 while others place the scalar element in the fourth position of the quaternion vector as is the case here.

three units i, j, k , which satisfy

$$\begin{aligned} i^2 &= j^2 = k^2 = -1, \\ ij &= -ji = k, \\ jk &= -kj = i, \quad \text{and} \\ ki &= -ik = j, \end{aligned} \quad (2.35)$$

with the fourth unit being the real number $+1$. Once again, let F_β be attached to the rotating body and F_α be the reference system. If I_β^α rotates R^3 through an angle γ about the axis ξ_α to bring F_α into alignment with F_β , then the unit quaternion defined by

$$\mathbf{q} = \left[\xi_{\alpha,1} \sin \frac{\gamma}{2}, \xi_{\alpha,2} \sin \frac{\gamma}{2}, \xi_{\alpha,3} \sin \frac{\gamma}{2}, \cos \frac{\gamma}{2} \right]^T \quad (2.36)$$

may be used to generate the rotation matrix I_β^α through the relationship

$$I_\beta^\alpha = \begin{bmatrix} q_1^2 - q_2^2 - q_3^2 + q_4^2 & 2(q_1q_2 - q_3q_4) & 2(q_1q_3 + q_2q_4) \\ 2(q_1q_2 + q_3q_4) & -q_1^2 + q_2^2 - q_3^2 + q_4^2 & 2(q_2q_3 - q_1q_4) \\ 2(q_1q_3 - q_2q_4) & 2(q_2q_3 + q_1q_4) & -q_1^2 - q_2^2 + q_3^2 + q_4^2 \end{bmatrix}. \quad (2.37)$$

The quaternion $\mathbf{q}(t)$ propagates in time according to the differential equation

$$\dot{\mathbf{q}}(t) = \Omega[\mathbf{w}(t)]\mathbf{q}(t), \quad (2.38)$$

where $\mathbf{w} \equiv [\mathbf{w}_\alpha^\beta]_\alpha = [w_1, w_2, w_3]^T$ is the angular velocity vector of F_β with respect to F_α expressed in F_α and

$$\Omega[\mathbf{w}] = \frac{1}{2} \begin{bmatrix} 0 & -w_3 & w_2 & -w_1 \\ w_3 & 0 & -w_1 & -w_2 \\ -w_2 & w_1 & 0 & -w_3 \\ w_1 & w_2 & w_3 & 0 \end{bmatrix}. \quad (2.39)$$

With constant angular velocity, the solution of (2.38) is simply

$$\mathbf{q}(t) = \exp[\Omega \cdot (t - t_0)] \mathbf{q}(t_0). \quad (2.40)$$

It can be verified by direct computation that

$$\Omega^2 = -\frac{1}{4} \|\mathbf{w}\|^2 \mathbf{I}_4, \quad (2.41)$$

which implies that the highest power of Ω in a power series is one. Expansion of (2.40) followed by repeated application of (2.41) gives

$$\mathbf{q}(t) = \left[\cos(\|\mathbf{w}\| (t - t_0)/2) \mathbf{I}_4 + \frac{2}{\|\mathbf{w}\|} \sin(\|\mathbf{w}\| (t - t_0)/2) \Omega \right] \mathbf{q}(t_0). \quad (2.42)$$

If, in addition, F_α and F_β are aligned at $t = t_0 = 0$, then $\mathbf{q}(t_0) = [0, 0, 0, 1]^T$ and (2.42) can be written as in (2.36) with

$$\begin{aligned} \xi_\alpha &= \frac{\mathbf{w}}{\|\mathbf{w}\|}, \text{ and} \\ \gamma &= \|\mathbf{w}\| t. \end{aligned} \quad (2.43)$$

Equation (2.43) is only valid when $\mathbf{w}$ is constant. Closed form solutions for $\mathbf{q}(t)$ are also available [30] for rotation with constant precession.

In the above discussions, F_α is the reference system while F_β is attached to the rotating body. Matrices denoted by $\mathbf{I}_\beta^\alpha$, therefore, represent change-of-basis transformations which may be applied to vectors in F_α to obtain representations for these vectors in F_β . In most of the applications considered in subsequent chapters, vectors representing positions of feature points expressed in an object-centred system (F_β) must be transformed into their representation in the reference system (F_α). This computation requires the inverse transformation, $\mathbf{I}_\alpha^\beta = (\mathbf{I}_\beta^\alpha)^T$, of those given in (2.11), (2.12), (2.17), and (2.37) above.

In the evaluation of state estimation algorithms considered in subsequent chapters, it is convenient to define a convergence index for these rotational operators. The roll-pitch-yaw and exponential formulations provide intuitive parameterizations for the three-dimensional rotational group. However, the quaternion expression is somewhat more abstract in that it is difficult to grasp the significance of a quaternion estimation error; for example, to say that an estimated quaternion $\hat{\mathbf{q}}$ has an error $\tilde{\mathbf{q}}$, what does this mean in terms of the error in the corresponding rotational operator $\tilde{\mathbf{I}}_\alpha^\beta \triangleq \mathbf{I}_\alpha^\beta - \hat{\mathbf{I}}_\alpha^\beta$. A convenient measure of convergence at the k th time step is given by the scaled Frobenius norm

$$J(k) \triangleq \frac{1}{2\sqrt{3}} \left(\text{Tr}(\tilde{\mathbf{I}}_\alpha^\beta(k)(\tilde{\mathbf{I}}_\alpha^\beta(k))^T) \right)^{\frac{1}{2}}. \quad (2.44)$$

The Frobenius matrix norm without the normalization factor $1/2\sqrt{3}$ has been employed in investigations such as [47]. Because $\mathbf{I}_\alpha^p(k)$ is orthogonal, and if $\hat{\mathbf{I}}_\alpha^p(k)$ is also orthogonal, the scale factor employed in (2.44) implies that the smallest interval containing $J(k)$ is $[0, 1]$.

2.3 Kalman Filtering Review

Applications of Kalman filtering to trajectory estimation problems has received widespread attention since the introduction of Kalman's sequential state estimation technique [43]. Chang and Tabaczynski [45] give a thorough discussion of many important aspects of Kalman filtering approaches for target tracking problems. In general, the trajectory estimation problem is one of nonlinear estimation for which the optimal (conditional mean) nonlinear estimator cannot be realized with a finite-dimensional implementation. Furthermore, the mathematical model representing the target trajectory is seldom exact. Consequently, most practical implementations of nonlinear tracking filters are suboptimal.

The Extended Kalman Filter (EKF) is a suboptimal approach which applies linear Kalman filtering theory to a linear approximation of the nonlinear problem. A rigorous development of filtering equations requires the use of stochastic integral and differential equations and mathematical expectation conditioned on σ -algebras generated by the observation process, and formulates solutions through Ito calculus ([51, 58, for example]). In applications, however, the somewhat more intuitive approaches using "formal" manipulations of white noise processes and expectation conditioned on sets of observations are often preferred [45], [47], [19]–[36]. This approach will also be taken in the following review of the EKF formulation. The following subsections draw primarily from [51]–[55].

In many cases, the target tracking problem can be formulated in terms of discrete-time system (plant) and measurement models. In the case of image-based tracking, measurements are taken from image frames which arrive at discrete instants in time. The measurement model for this class of problems, therefore, is always discrete. In addition, the perspective projection image formation model results in a nonlinear

measurement model. If the system model is continuous in time, it can often be discretized, or the "continuous-discrete" [51] form of the filter can be employed.

Notation and filtering equations for the discrete-time linear state estimation problem are reviewed in Section 2.3.1, with extensions to approximate filters for the nonlinear discrete-time problem in Section 2.3.2 and the continuous-discrete form of extended Kalman filtering in Section 2.3.3. Finally, Section 2.3.4 describes local iterations which enhance filter performance in the presence of nonlinear measurement models.

2.3.1 Discrete-Time Kalman Filter

Consider a linear discrete-time dynamic system model given by

$$\mathbf{x}(k+1) = \Phi(k)\mathbf{x}(k) + \mathbf{B}(k)\mathbf{u}(k) + \mathbf{G}(k)\mathbf{w}(k), \quad (2.45)$$

where

$$\begin{aligned} k &\sim \text{integer valued sample index, time } t = kT \text{ where} \\ &\quad T \text{ is the sample period,} \\ \mathbf{x} \in R^n &\sim \text{vector of state variables,} \\ \mathbf{u} \in R^q &\sim \text{deterministic input vector,} \\ \mathbf{w} \in R^p &\sim \text{zero mean, white, Gaussian, system noise} \\ &\quad \text{process with covariance matrix } \mathbf{Q}(k), \\ \Phi \in R^{n \times n} &\sim \text{state transition matrix,} \\ \mathbf{B} \in R^{n \times q} &\sim \text{input matrix mapping the input vector} \\ &\quad \mathbf{u} \text{ into the state space, and} \\ \mathbf{G} \in R^{n \times p} &\sim \text{matrix mapping the noise process} \\ &\quad \mathbf{w} \text{ into the state space.} \end{aligned} \quad (2.46)$$

The linear, discrete-time measurement model is given by

$$\mathbf{z}(k) = \mathbf{H}(k)\mathbf{x}(k) + \mathbf{v}(k), \quad (2.47)$$

where

$$\begin{aligned}
 \mathbf{z} \in R^m &\sim \text{vector of observations or measurements,} \\
 \mathbf{H} \in R^{n \times m} &\sim \text{measurement matrix relating state variables} \\
 &\quad \text{to measurement variables, and} \\
 \mathbf{v} \in R^m &\sim \text{zero mean, white, Gaussian, measurement noise} \\
 &\quad \text{process with covariance matrix } \mathbf{R}(k).
 \end{aligned} \tag{2.48}$$

With definitions (2.46) and (2.48), the system and measurement noise processes satisfy

$$\begin{aligned}
 \mathcal{E}\{\mathbf{w}(k)\} &= 0 & \mathcal{E}\{\mathbf{w}(k)\mathbf{w}^T(j)\} &= \mathbf{Q}(k)\delta_{kj}, \\
 \mathcal{E}\{\mathbf{v}(k)\} &= 0 & \mathcal{E}\{\mathbf{v}(k)\mathbf{v}^T(j)\} &= \mathbf{R}(k)\delta_{kj},
 \end{aligned} \tag{2.49}$$

where $\mathcal{E}\{\cdot\}$ denotes expectation and δ_{jk} is the Kronecker delta product. In the present analysis, $\mathbf{w}$ and $\mathbf{v}$ are also assumed to be uncorrelated, i.e.,

$$\mathcal{E}\{\mathbf{w}(k)\mathbf{v}^T(j)\} = 0, \tag{2.50}$$

for all integers j and k .

The expectation of the state $\mathbf{x}$ at time step k conditioned on all measurements up to time step j is denoted by

$$\hat{\mathbf{x}}(k|j) \triangleq \mathcal{E}\{\mathbf{x}(k)|Z_j\} \tag{2.51}$$

where $Z_j = \{\mathbf{z}(i), i = 1, 2, \dots, j\}$ represents the set of all measurements up to and including the j th time step. It is precisely the conditional mean in (2.51), usually with $j = k$, which the optimal filter is required to estimate. In the case of linear system and measurement models, assuming exact mathematical models and characterization of the noise processes, the Kalman filter provides a recursive "formula" for this optimal estimate. It is well known that in this case the Kalman filter provides an unbiased, minimum variance, consistent estimate $\hat{\mathbf{x}}(k|k)$, which is also the maximum likelihood estimate of the true state $\mathbf{x}(k)$.

The error between the filter state and the true state is given by

$$\tilde{\mathbf{x}}(k|j) \triangleq \mathbf{x}(k) - \hat{\mathbf{x}}(k|j), \tag{2.52}$$

with covariance matrix

$$\mathbf{P}(k|j) \triangleq \mathcal{E} \{ \hat{\mathbf{x}}(k|j) \hat{\mathbf{x}}^T(k|j) \}. \quad (2.53)$$

The development of the filtering equations assumes that the true initial state of the system $\mathbf{x}(0) \sim N(\mathbf{x}_0, \mathbf{P}_0)$ ³. Throughout the remainder of this work, it is assumed that an initial estimate, $\hat{\mathbf{x}}(0|-1)$, of the mean of the true initial state, $\mathcal{E}\{\mathbf{x}(0)\}$, is available just prior to taking initial measurements, $\mathbf{z}(0)$. In theory, then, the recursive filter is initialized with

$$\begin{aligned} \hat{\mathbf{x}}(0|-1) &= \mathcal{E}\{\mathbf{x}(0)\} \triangleq \mathbf{x}_0, \text{ and} \\ \mathbf{P}(0|-1) &= \mathcal{E}\{\hat{\mathbf{x}}(0|-1) \hat{\mathbf{x}}^T(0|-1)\} \triangleq \mathbf{P}_0. \end{aligned} \quad (2.54)$$

In practice, however, the filter is often initialized with an unbiased estimator of $\mathbf{x}_0$ for which some *a priori* statistical information is known of the form

$$\hat{\mathbf{x}}(0|-1) \sim N(\mathbf{x}_0, \mathbf{P}_0). \quad (2.55)$$

After $k-1$ measurements have been taken, the recursive equations of the linear Kalman filter are summarized as follows:

1. State and covariance prediction (or time update) -

$$\begin{aligned} \hat{\mathbf{x}}(k|k-1) &= \Phi(k-1) \hat{\mathbf{x}}(k-1|k-1) + \mathbf{B}(k-1) \mathbf{u}(k-1), \text{ and} \\ \mathbf{P}(k|k-1) &= \Phi(k-1) \mathbf{P}(k-1|k-1) \Phi^T(k-1) \\ &\quad + \mathbf{G}(k-1) \mathbf{Q}(k-1) \mathbf{G}^T(k-1). \end{aligned} \quad (2.56)$$

2. Measurement update computed following the k th measurement -

$$\begin{aligned} \mathbf{K}(k) &= \mathbf{P}(k|k-1) \mathbf{H}^T(k) \\ &\quad \cdot [\mathbf{H}(k) \mathbf{P}(k|k-1) \mathbf{H}^T(k) + \mathbf{R}(k)]^{-1}, \\ \hat{\mathbf{x}}(k|k) &= \hat{\mathbf{x}}(k|k-1) + \mathbf{K}(k) [\mathbf{z}(k) - \mathbf{H}(k) \hat{\mathbf{x}}(k|k-1)], \text{ and} \\ \mathbf{P}(k|k) &= [\mathbf{I} - \mathbf{K}(k) \mathbf{H}(k)] \mathbf{P}(k|k-1). \end{aligned} \quad (2.57)$$

³The notation $N(\mathbf{m}, \mathbf{C})$ denotes a normally distributed random variable with mean $\mathbf{m}$ and covariance $\mathbf{C}$.

An equivalent form, known as the "information form", of the measurement update stage of the filter, which is often used in place of (2.57), is given by

$$\begin{aligned} \mathbf{P}^{-1}(k|k) &= \mathbf{P}^{-1}(k|k-1) + \mathbf{H}^T(k)\mathbf{R}^{-1}(k)\mathbf{H}(k), \\ \mathbf{K}(k) &= \mathbf{P}(k|k)\mathbf{H}^T(k)\mathbf{R}^{-1}(k), \text{ and} \\ \hat{\mathbf{x}}(k|k) &= \hat{\mathbf{x}}(k|k-1) + \mathbf{K}(k)[\mathbf{z}(k) - \mathbf{H}(k)\hat{\mathbf{x}}(k|k-1)]. \end{aligned} \quad (2.58)$$

In any implementation of the above filtering equations, the following parameters must be specified in addition to the measurement sequence: 1) an initial state estimate $\hat{\mathbf{x}}(0|-1)$; 2) an estimate of the initial error covariance $\mathbf{P}_0$; 3) the system or plant noise covariances $\mathbf{Q}(k)$; and 4) the measurement or observation noise covariances $\mathbf{R}(k)$. The initial state estimate and its error covariance matrix often result from a batch estimation technique applied to the first few image frames [26, 28, 32]. In simulations, the measurement noise covariance is known, however, in applications to real imagery, determination of the actual measurement noise is difficult since this involves modeling of various sources of error in the imaging system and feature detection algorithms. Specification of the system noise covariance $\mathbf{Q}(k)$ is seldom exact in object tracking problems since unmodeled dynamics resulting from manoeuvring object trajectories are difficult to quantify. As a result, $\mathbf{Q}(k)$ together with the matrix $\mathbf{G}(k)$ are often treated as "tuning" parameters which are selected to give satisfactory convergent behavior of the filter over a wide range of applications.

In Equation (2.56) the previous state estimate $\hat{\mathbf{x}}(k-1|k-1)$ is propagated in time between measurements according to the noise-free dynamic system model to obtain a state estimate $\hat{\mathbf{x}}(k|k-1)$ just prior to a measurement event. Once measurements are available, the state estimate is corrected in (2.57) by an amount proportional to the current error $[\mathbf{z}(k) - \mathbf{H}(k)\hat{\mathbf{x}}(k|k-1)]$ called the innovation. The proportionality factor $\mathbf{K}(k)$ is called the Kalman gain which, from (2.58), is proportional to the uncertainty in the state estimate $\mathbf{P}(k|k)$ and inversely proportional to the measurement uncertainty $\mathbf{R}(k)$. The system noise covariance $\mathbf{Q}(k)$ which is mapped to the state space through $\mathbf{G}(k)$ adds a positive definite matrix to the filter error covariance matrix during each time update (2.56) which reflects a "diffusion" of confidence in the state estimate over the time interval between measurements. In

the absence of system noise ($Q \equiv 0$) the error covariance and gain matrices may decrease monotonically in the presence of measurement data, depending on the state transition matrix through the similarity transformation in (2.56), i.e., consider (2.58) for the scalar case with $H \equiv 1$. In this case, the filter eventually neglects further measurements and operates solely on the system model which, with $Q \equiv 0$, is assumed to be exact.

2.3.2 Discrete-Time Extended Kalman Filter

The problem of estimating the state of a nonlinear system can be formulated as an extension of mean square estimation to dynamic systems. Formally, the method for deriving the fundamental equations of the optimal filter is quite similar to that used to design the optimal linear filter (Kalman filter) for linear systems with Gaussian noise processes. However, in the nonlinear case, the explicit analytical solution cannot, in general, be found and hence approximate numerical solutions are often considered.

In the case of linear systems, all of the variables and processes involved (states, noises, measurements, estimates, and errors) are usually assumed to be normally distributed. Therefore, mean values and covariance matrices are "sufficient statistics." In the nonlinear case, on the other hand, the state, observation, and estimation processes are not, in general, normally distributed even if the input noise sources are. As a result, the entire probability distribution may be required for optimal filter design. In addition, the filter performance and parameters are often dependent on the filter estimate so that it is even more difficult to obtain bounds for filter performance.

Consider the nonlinear system model given by

$$\mathbf{x}(k+1) = \mathbf{f}[\mathbf{x}(k)] + \mathbf{B}(k)\mathbf{u}(k) + \mathbf{g}[\mathbf{x}(k)]\mathbf{w}(k), \quad (2.59)$$

and the nonlinear measurement model

$$\mathbf{z}(k) = \mathbf{h}[\mathbf{x}(k)] + \mathbf{v}(k), \quad (2.60)$$

where $f : R^n \mapsto R^n$, $g : R^n \mapsto (R^{n \times p})$, and $h : R^n \mapsto R^m$ are, in general, nonlinear functions of the state vector $\mathbf{x}(k)$. The mapping of the input vector $\mathbf{u}$ to the state space may also be nonlinear, but this case will not be considered here and, in any event, does not alter the filter structure significantly.

The Extended Kalman Filter (EKF) is based on the concept that if the nonlinearities in f and h are sufficiently smooth, these functions can be expanded in a Taylor series and approximated by linear terms. Furthermore, if the filter works properly, it will provide a reasonable estimate of the conditional mean $\mathcal{E}\{\mathbf{x}(k)|Z_k\}$ which is what the optimal filter is required to estimate. For this reason, the Taylor series expansion of nonlinear terms is formulated about the most recent state estimate. The primary consequence of this "feedback" of the estimate into the model is that the EKF is a nonlinear estimation technique.

With the definitions

$$\Phi(k) \triangleq \left. \frac{\partial f[\mathbf{x}]}{\partial \mathbf{x}} \right|_{\mathbf{x}=\hat{\mathbf{x}}(k|k)}, \quad (2.61)$$

$$\mathbf{G}(k) \triangleq \mathbf{g}[\hat{\mathbf{x}}(k|k)], \quad \text{and} \quad (2.62)$$

$$\mathbf{H}(k) \triangleq \left. \frac{\partial h[\mathbf{x}]}{\partial \mathbf{x}} \right|_{\mathbf{x}=\hat{\mathbf{x}}(k|k-1)}, \quad (2.63)$$

and neglecting all terms beyond first order in the Taylor series expansions, approximate linear system and measurement models can be written, respectively, as

$$\mathbf{x}(k+1) = \Phi(k)\mathbf{x}(k) + \tilde{\mathbf{u}}(k) + \mathbf{G}(k)\mathbf{w}(k), \quad \text{and} \quad (2.64)$$

$$\mathbf{z}(k) = \mathbf{H}(k)\mathbf{x}(k) + \mathbf{y}(k) + \mathbf{v}(k), \quad (2.65)$$

where

$$\tilde{\mathbf{u}}(k) \triangleq \mathbf{f}[\hat{\mathbf{x}}(k|k)] - \Phi(k)\hat{\mathbf{x}}(k|k) + \mathbf{B}(k)\mathbf{u}(k), \quad \text{and} \quad (2.66)$$

$$\mathbf{y}(k) \triangleq \mathbf{h}[\hat{\mathbf{x}}(k|k-1)] - \mathbf{H}(k)\hat{\mathbf{x}}(k|k-1). \quad (2.67)$$

The EKF equations are formulated, essentially, as the standard Kalman filtering equations for the linear system given by (2.61)-(2.67).

After $k-1$ measurements have been taken, the recursive equations of the extended Kalman filter are summarized as follows:

1. State and covariance prediction (or time update) -

$$\begin{aligned}\hat{\mathbf{x}}(k|k-1) &= \mathbf{f}[\hat{\mathbf{x}}(k-1|k-1)] + \mathbf{B}(k)\mathbf{u}(k), \text{ and} \\ \hat{\mathbf{P}}(k|k-1) &= \Phi(k-1)\hat{\mathbf{P}}(k-1|k-1)\Phi^T(k-1) \\ &\quad + \mathbf{G}(k-1)\mathbf{Q}(k-1)\mathbf{G}^T(k-1).\end{aligned}\tag{2.68}$$

2. Measurement update computed following the k th measurement -

$$\begin{aligned}\mathbf{K}(k) &= \hat{\mathbf{P}}(k|k-1)\mathbf{H}^T(k) \\ &\quad \cdot [\mathbf{H}(k)\hat{\mathbf{P}}(k|k-1)\mathbf{H}^T(k) + \mathbf{R}(k)]^{-1}, \\ \hat{\mathbf{x}}(k|k) &= \hat{\mathbf{x}}(k|k-1) + \mathbf{K}(k) \{ \mathbf{z}(k) - \mathbf{h}[\hat{\mathbf{x}}(k|k-1)] \}, \text{ and} \\ \hat{\mathbf{P}}(k|k) &= [\mathbf{I} - \mathbf{K}(k)\mathbf{H}(k)] \hat{\mathbf{P}}(k|k-1).\end{aligned}\tag{2.69}$$

The accent on $\hat{\mathbf{P}}$ has been used to emphasize the fact that the covariance estimate is dependent on the filter estimate.

2.3.3 Continuous-Discrete Extended Kalman Filter

In Section 2.3.2, both the system and measurement models were assumed to be available in discrete-time form. In general, however, the system model is often developed in continuous-time form, while measurements are taken at discrete instants in time.

Consider a continuous-time system which is described by the stochastic differential equation

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}, t) + \mathbf{G}(t)\mathbf{w}(t), \quad t \geq 0, \quad \mathbf{x}(0) \sim N(\bar{\mathbf{x}}_0, \mathbf{P}_0),\tag{2.70}$$

where $\mathbf{w}(t)$ is a zero-mean white Gaussian noise process with

$$\mathcal{E} \{ \mathbf{w}(t)\mathbf{w}^T(\tau) \} = \mathbf{Q}(t)\delta(t - \tau),\tag{2.71}$$

and $\delta(t - \tau)$ is the Dirac-delta function. The system model may also depend on a deterministic input vector $\mathbf{u}(t)$. To simplify the notation, this dependence is not shown explicitly here and, in any event, does not alter the filter structure significantly. As in the discrete-time case of Section 2.3.2, the nonlinear system model

of (2.70) is linearized approximately through a first-order Taylor series expansion about a nominal trajectory $\hat{\mathbf{x}}(t)$ which satisfies

$$\dot{\hat{\mathbf{x}}}(t) = \mathbf{f}(\hat{\mathbf{x}}, t), \quad t \geq 0. \quad (2.72)$$

The state perturbation is defined as

$$\tilde{\mathbf{x}}(t) \triangleq \mathbf{x}(t) - \hat{\mathbf{x}}(t), \quad (2.73)$$

which satisfies the stochastic differential equation

$$\begin{aligned} \dot{\tilde{\mathbf{x}}}(t) &= \mathbf{f}(\mathbf{x}, t) - \mathbf{f}(\hat{\mathbf{x}}, t) + \mathbf{G}(t)\mathbf{w}(t) \\ &\approx \mathbf{F}[t; \hat{\mathbf{x}}]\tilde{\mathbf{x}}(t) + \mathbf{G}(t)\mathbf{w}(t), \end{aligned} \quad (2.74)$$

with the initial condition $\tilde{\mathbf{x}}(0) \sim N(\mathbf{x}_0 - \hat{\mathbf{x}}(0), \mathbf{P}_0)$ and where

$$\mathbf{F}[t; \hat{\mathbf{x}}] \triangleq \left. \frac{\partial \mathbf{f}(\mathbf{x}, t)}{\partial \mathbf{x}} \right|_{\mathbf{x}=\hat{\mathbf{x}}}. \quad (2.75)$$

It should be noted that $\mathbf{F}$ is evaluated at particular values of $\hat{\mathbf{x}}(t)$ and hence is a function of t only. Specifically, as in the discrete-time case of Section 2.3.2, the nominal trajectory is taken as the most recent state estimate which, for the linearized system model, is given by

$$\hat{\mathbf{x}}(t_k) \equiv \hat{\mathbf{x}}(k|k). \quad (2.76)$$

In principle, equation (2.74) can be written in discrete-time form as

$$\tilde{\mathbf{x}}(k+1) = \hat{\Phi}(t_{k+1}, t_k)\tilde{\mathbf{x}}(k) + \mathbf{w}_d(k+1), \quad (2.77)$$

where $\hat{\Phi}(t_{k+1}, t_k)$ is the state transition matrix which satisfies [51]

$$\begin{aligned} \frac{d\hat{\Phi}(t, \tau)}{dt} &= \mathbf{F}[t; \mathbf{x}(t)]\hat{\Phi}(t, \tau), \\ \hat{\Phi}(\tau, \tau) &= \mathbf{I}_n, \quad \forall \tau, \\ \hat{\Phi}(t, \tau)\hat{\Phi}(\tau, \zeta) &= \hat{\Phi}(t, \zeta), \quad \forall t, \tau, \zeta, \end{aligned} \quad (2.78)$$

and $\mathbf{w}_d(k+1)$ is a zero-mean, white Gaussian sequence, $\mathbf{w}_d(k+1) \sim N(0, \mathbf{Q}_d(k+1))$, where

$$\mathbf{Q}_d(k+1) = \int_{t_k}^{t_{k+1}} \hat{\Phi}(t_{k+1}, \tau)\mathbf{G}(\tau)\mathbf{Q}(\tau)\mathbf{G}^T(\tau)\hat{\Phi}^T(t_{k+1}, \tau) d\tau. \quad (2.79)$$

If $\mathbf{F}$ is approximately constant over the interval $(t_k, t_{k+1}]$, where $t_{k+1} - t_k = T_k$, the state transition matrix for each sample interval can be estimated as

$$\hat{\Phi}(t_{k+1}, t_k) = \exp(\mathbf{F}T_k). \quad (2.80)$$

In general, however, the computation of the state transition matrix is often difficult and, therefore, replaced by numerical integration of the Riccati covariance equation [51]

$$\begin{aligned} \dot{\hat{\mathbf{P}}}(t|t_k) &= \mathbf{F}[t; \hat{\mathbf{x}}(t|t_k)]\hat{\mathbf{P}}(t|t_k) \\ &+ \hat{\mathbf{P}}(t|t_k)\mathbf{F}^T[t; \hat{\mathbf{x}}(t|t_k)] + \mathbf{G}(t)\mathbf{Q}(t)\mathbf{G}^T(t). \end{aligned} \quad (2.81)$$

The nonlinear, discrete-time measurement model is again given by (2.60) and is treated in a manner identical to that used in Section 2.3.2. The extended Kalman filter is then formulated based on the approximate linear perturbation equation (2.74), or its discrete form (2.77), and the linear approximation of the measurement model (2.65). However, in this case, the state prediction is computed by integration of the nonlinear noise-free system differential equation. After $k - 1$ measurements have been taken, the recursive equations of the continuous-discrete extended Kalman filter are summarized as follows:

1. State and covariance prediction (or time update) - Either

$$\begin{aligned} \hat{\mathbf{x}}(k|k-1) &= \hat{\mathbf{x}}(k-1|k-1) + \int_{t_{k-1}}^{t_k} \mathbf{f}[\hat{\mathbf{x}}(t|t_{k-1}), t] dt \quad \text{and} \quad (2.82) \\ \hat{\mathbf{P}}(k|k-1) &= \hat{\mathbf{P}}(k-1|k-1) + \int_{t_{k-1}}^{t_k} \dot{\hat{\mathbf{P}}}[t; \hat{\mathbf{x}}(t|t_{k-1})] dt, \end{aligned}$$

where $\dot{\hat{\mathbf{P}}}[t; \mathbf{x}(t|t_{k-1})]$ is defined in (2.81), or equivalently,

$$\begin{aligned} \hat{\mathbf{x}}(k|k-1) &= \hat{\Phi}(k-1)\hat{\mathbf{x}}(k-1|k-1) \\ \hat{\mathbf{P}}(k|k-1) &= \hat{\Phi}(k-1)\hat{\mathbf{P}}(k-1|k-1)\hat{\Phi}^T(k-1) + \mathbf{Q}_d(k). \end{aligned} \quad (2.83)$$

2. Measurement update computed following the k th measurement -

$$\begin{aligned} \mathbf{K}(k) &= \hat{\mathbf{P}}(k|k-1)\mathbf{H}^T(k) \\ &\cdot [\mathbf{H}(k)\hat{\mathbf{P}}(k|k-1)\mathbf{H}^T(k) + \mathbf{R}(k)]^{-1}, \quad (2.84) \\ \hat{\mathbf{x}}(k|k) &= \hat{\mathbf{x}}(k|k-1) + \mathbf{K}(k) \{ \mathbf{z}(k) - \mathbf{h}[\hat{\mathbf{x}}(k|k-1)] \}, \text{ and} \\ \hat{\mathbf{P}}(k|k) &= [\mathbf{I} - \mathbf{K}(k)\mathbf{H}(k)]\hat{\mathbf{P}}(k|k-1). \end{aligned}$$

Again, the accent on $\hat{\mathbf{P}}$ has been used to emphasize the fact that the covariance estimate is dependent on the filter estimate.

2.3.4 Local Iterations

The Iterated Extended Kalman Filter (IEKF) is designed to reduce the effect of measurement nonlinearities on the performance of the linear filter applied to a linearized nonlinear system [51, p. 279]. Between observations the conditional mean and covariance matrix propagate according to the equations of (2.68). Following the time update of the state and covariance from time step $k - 1$ to time step k , and with the k th measurement available, the iteration is initialized with

$$\hat{\mathbf{x}}(k|k)_0 = \hat{\mathbf{x}}(k|k - 1). \quad (2.85)$$

The iterations are carried out on the measurement update equations in (2.69) according to the following algorithm:

1. Linearize the measurement equations about the current iterate -

$$\mathbf{H}(k)_n = \left. \frac{\partial h[\mathbf{x}]}{\partial \mathbf{x}} \right|_{\mathbf{x}=\hat{\mathbf{x}}(k|k)_n}, \quad (2.86)$$

2. Compute the Kalman gain -

$$\begin{aligned} \mathbf{K}(k)_n &= \hat{\mathbf{P}}(k|k - 1) \mathbf{H}^T(k)_n \\ &\cdot [\mathbf{H}(k)_n \hat{\mathbf{P}}(k|k - 1) \mathbf{H}^T(k)_n + \mathbf{R}(k)]^{-1}, \end{aligned} \quad (2.87)$$

3. Compute the measurement update -

$$\begin{aligned} \hat{\mathbf{x}}(k|k)_{n+1} &= \hat{\mathbf{x}}(k|k - 1) + \mathbf{K}(k)_n \{z(k) - h[\hat{\mathbf{x}}(k|k)_n] \\ &\quad - \mathbf{H}(k)_n [\hat{\mathbf{x}}(k|k - 1) - \hat{\mathbf{x}}(k|k)_n]\}. \end{aligned} \quad (2.88)$$

Iteration is terminated when there is no significant change in consecutive iterates. Then the last iterate, $\hat{\mathbf{x}}(k|k)_N$ say, is taken for the estimate $\hat{\mathbf{x}}(k|k)$ and the covariance matrix is computed based on the linearization about $\hat{\mathbf{x}}(k|k)_N$,

$$\hat{\mathbf{P}}(k|k) = [\mathbf{I} - \mathbf{K}(k)_N \mathbf{H}(k)_N] \hat{\mathbf{P}}(k|k - 1). \quad (2.89)$$

The iterative procedure described by (2.86)-(2.88) can be interpreted as a modified Newton-Raphson search for the conditional mode of the posterior density of the state estimate given the measurement at time step k , assuming the prior density is Gaussian [51, pp. 349-351]. The mode of the posterior density is then used for the conditional mean. In nonlinear problems, these local iterations can be expected to produce a biased estimate since the mode is not, in general, equivalent to the mean. However, as the error variance becomes small, so does the bias in this estimate.

2.4 Cramer-Rao Covariance Bounds

It has been noted above that the EKF is generally a suboptimal state estimation technique. A natural consideration in any estimation problem is how well the proposed technique performs when compared to the optimal performance possible. A powerful result which provides a performance assessment of parameter estimation techniques is the Cramer-Rao inequality [56]. The following derivation draws from methods outlined in [44, 45, 51, 56, 57].

Consider a nonlinear discrete-time estimation problem with noise-free system model

$$\mathbf{x}(k+1) = \mathbf{f}[\mathbf{x}(k)] \quad (2.90)$$

and measurement model

$$\mathbf{z}(k) = \mathbf{h}[\mathbf{x}(k)] + \mathbf{v}(k), \quad (2.91)$$

where $\mathbf{v}(k) \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$ and is temporally white. Assume further that some unbiased *a priori* statistical information about $\mathbf{x}(0)$ is available of the form

$$\mathbf{x}_0 \sim \mathcal{N}(\mathbf{x}(0), \mathbf{P}_0), \quad (2.92)$$

where $\mathbf{x}_0$ and $\mathbf{v}(k)$ are independent. The objective is to determine how well $\mathbf{x}(N)$, $N \geq 0$, can be estimated given the sequence of observations Z_N^* , where

$$Z_N^* \triangleq \{\mathbf{x}_0, \mathbf{z}(0), \mathbf{z}(1), \dots, \mathbf{z}(N)\}. \quad (2.93)$$

Under the stated assumptions of independence and normal distributions, the joint conditional probability density function of Z_N^* given $\mathbf{x}(N)$ can be written,

from (2.91) and (2.92), as

$$p_N \triangleq p_{Z_N^*|x(N)} = \frac{1}{A} e^{-\frac{1}{2} \Psi[Z_N^*; x(N)]}, \quad (2.94)$$

where⁴

$$A \triangleq (2\pi)^{n/2} |\mathbf{P}_0|^{1/2} \prod_{k=0}^N [(2\pi)^{m/2} |\mathbf{R}(k)|^{1/2}], \quad (2.95)$$

$$\begin{aligned} \Psi[Z_N^*; x(N)] &\triangleq (\mathbf{x}_0 - \hat{\mathbf{x}}_0)^T \mathbf{P}_0^{-1} (\mathbf{x}_0 - \hat{\mathbf{x}}_0) \\ &\quad + \sum_{k=0}^N [\mathbf{z}(k) - \hat{\mathbf{z}}(k)]^T \mathbf{R}^{-1}(k) [\mathbf{z}(k) - \hat{\mathbf{z}}(k)], \end{aligned} \quad (2.96)$$

and

$$\hat{\mathbf{x}}_0 \triangleq \mathcal{E}\{\mathbf{x}_0\} = \mathbf{x}(0), \text{ and} \quad (2.97)$$

$$\hat{\mathbf{z}}(k) \triangleq \mathcal{E}\{\mathbf{z}(k)\} = \mathbf{h}[\mathbf{x}(k)]. \quad (2.98)$$

The Cramer-Rao lower bound for the error covariance matrix $\mathbf{S}(N)$ of an unbiased estimator $\hat{\mathbf{x}}(N)$ of the state $\mathbf{x}(N)$, where the estimate is based on the observation sequence Z_N^* , is given by

$$\mathbf{S}(N) \triangleq \mathcal{E}\{[\hat{\mathbf{x}}(N) - \mathbf{x}(N)][\hat{\mathbf{x}}(N) - \mathbf{x}(N)]^T\} \geq \mathbf{J}^{-1}(N), \quad (2.99)$$

where $\mathbf{J}(N)$ is Fisher's information matrix. The matrix inequality $\mathbf{S} \geq \mathbf{J}^{-1}$ is equivalent to stating that $(\mathbf{S} - \mathbf{J}^{-1})$ is positive semi-definite. In particular, since the diagonal elements of a positive semi-definite matrix are non-negative, the diagonal elements of $\mathbf{J}^{-1}(N)$ provide the estimation error variance lower bounds for the corresponding elements of $\hat{\mathbf{x}}(N)$. In the linear system and measurement case, (2.45) and (2.47), the discrete Kalman filter given by (2.56) and (2.57) provides the minimum variance unbiased estimate. If the system noise $\mathbf{w}(k)$ in (2.45) is zero, then for the linear case

$$\mathbf{S}(N) = \mathbf{P}(N|N) = \mathbf{J}^{-1}(N). \quad (2.100)$$

In the nonlinear EKF formulation, it is evident that $\mathbf{S}(N)$ and $\hat{\mathbf{P}}(N|N)$ are not, in general, equivalent.

⁴The notation $|\mathbf{P}|$ denotes the determinant of $\mathbf{P}$.

Fisher's information measure on $\mathbf{x}(N)$ contained in Z_N^* is given by two equivalent representations [57, pp. 91-93]:

$$\mathbf{J}(N) = \mathcal{E} \left\{ \left(\frac{\partial \ln p_N}{\partial \mathbf{x}(N)} \right)^T \left(\frac{\partial \ln p_N}{\partial \mathbf{x}(N)} \right) \middle| \mathbf{x}(N) \right\}, \text{ or} \quad (2.101)$$

$$\mathbf{J}(N) = -\mathcal{E} \left\{ \left(\frac{\partial^2 \ln p_N}{\partial \mathbf{x}(N)^2} \right) \middle| \mathbf{x}(N) \right\}, \quad (2.102)$$

where the gradient $\partial \ln p_N / \partial \mathbf{x}(N)$ is a row vector. The inequality (2.99) with $\mathbf{J}(N)$ expressed as in (2.101) is termed the "Cramer-Rao inequality."

Taking the logarithm of p_N in (2.94) yields

$$\ln p_N = -\ln A - \frac{1}{2} \Psi[Z_N^*; \mathbf{x}(N)]. \quad (2.103)$$

The gradient of (2.103) is given by the row vector

$$\begin{aligned} \frac{\partial \ln p_N}{\partial \mathbf{x}(N)} &= (\mathbf{x}_0 - \bar{\mathbf{x}}_0)^T \mathbf{P}_0^{-1} \frac{\partial \bar{\mathbf{x}}_0}{\partial \mathbf{x}(N)} \\ &\quad + \sum_{k=0}^N [\mathbf{z}(k) - \bar{\mathbf{z}}(k)]^T \mathbf{R}^{-1}(k) \frac{\partial \bar{\mathbf{z}}(k)}{\partial \mathbf{x}(N)}. \end{aligned} \quad (2.104)$$

To evaluate (2.101), consider the four terms of the outer product of (2.104):

$$\begin{aligned} \left(\frac{\partial p_N}{\partial \mathbf{x}(N)} \right)^T \left(\frac{\partial p_N}{\partial \mathbf{x}(N)} \right) &= \\ &\left(\frac{\partial \bar{\mathbf{x}}_0}{\partial \mathbf{x}(N)} \right)^T \mathbf{P}_0^{-1} (\mathbf{x}_0 - \bar{\mathbf{x}}_0) (\mathbf{x}_0 - \bar{\mathbf{x}}_0)^T \mathbf{P}_0^{-1} \left(\frac{\partial \bar{\mathbf{x}}_0}{\partial \mathbf{x}(N)} \right) \\ &+ \sum_{k=0}^N \left(\frac{\partial \bar{\mathbf{x}}_0}{\partial \mathbf{x}(N)} \right)^T \mathbf{P}_0^{-1} (\mathbf{x}_0 - \bar{\mathbf{x}}_0) [\mathbf{z}(k) - \bar{\mathbf{z}}(k)]^T \mathbf{R}^{-1}(k) \left(\frac{\partial \bar{\mathbf{z}}(k)}{\partial \mathbf{x}(N)} \right) \\ &+ \sum_{k=0}^N \left(\frac{\partial \bar{\mathbf{z}}(k)}{\partial \mathbf{x}(N)} \right)^T \mathbf{R}^{-1}(k) [\mathbf{z}(k) - \bar{\mathbf{z}}(k)] (\mathbf{x}_0 - \bar{\mathbf{x}}_0)^T \mathbf{P}_0^{-1} \left(\frac{\partial \bar{\mathbf{x}}_0}{\partial \mathbf{x}(N)} \right) \\ &+ \left\{ \sum_{k=0}^N \left(\frac{\partial \bar{\mathbf{z}}(k)}{\partial \mathbf{x}(N)} \right)^T \mathbf{R}^{-1}(k) [\mathbf{z}(k) - \bar{\mathbf{z}}(k)] \right\} \\ &\quad \cdot \left\{ \sum_{j=0}^N [\mathbf{z}(j) - \bar{\mathbf{z}}(j)]^T \mathbf{R}^{-1}(j) \left(\frac{\partial \bar{\mathbf{z}}(j)}{\partial \mathbf{x}(N)} \right) \right\}. \end{aligned} \quad (2.105)$$

The conditional expectation in (2.101) can be computed from (2.105) with the following observations:

1. The Jacobian matrices $\partial \bar{\mathbf{x}}_0 / \partial \mathbf{x}(N)$ and $\partial \bar{\mathbf{z}}(k) / \partial \mathbf{x}(N)$ are deterministic;
2. From (2.91), (2.97), and (2.98), the random variables $(\mathbf{x}_0 - \bar{\mathbf{x}}_0)$ and $[\mathbf{z}(k) - \bar{\mathbf{z}}(k)]$, $k = 0, 1, \dots, N$, are independent with zero mean and hence the second and third terms of (2.105) do not contribute to the expectation and the product of summations in the last term of (2.105) reduces to a single summation of products with $k = j$;
3. By definitions (2.91) and (2.92),

$$\begin{aligned} \mathcal{E} \{ (\mathbf{x}_0 - \bar{\mathbf{x}}_0)(\mathbf{x}_0 - \bar{\mathbf{x}}_0)^T \} &= \mathbf{P}_0 \text{ and} \\ \mathcal{E} \{ [\mathbf{z}(k) - \bar{\mathbf{z}}(k)][\mathbf{z}(k) - \bar{\mathbf{z}}(k)]^T \} &= \mathbf{R}(k). \end{aligned}$$

Furthermore, from the definitions given in (2.97) and (2.98),

$$\frac{\partial \bar{\mathbf{x}}_0}{\partial \mathbf{x}(N)} = \frac{\partial \mathbf{x}(0)}{\partial \mathbf{x}(N)} \quad (2.106)$$

and

$$\begin{aligned} \frac{\partial \bar{\mathbf{z}}(k)}{\partial \mathbf{x}(N)} &= \frac{\partial \mathbf{h}[\mathbf{x}(k)]}{\partial \mathbf{x}(N)} \\ &= \mathbf{H}_t(k) \frac{\partial \mathbf{x}(k)}{\partial \mathbf{x}(N)}, \end{aligned} \quad (2.107)$$

where

$$\mathbf{H}_t(k) \triangleq \left. \frac{\partial \mathbf{h}[\mathbf{x}]}{\partial \mathbf{x}} \right|_{\mathbf{x}=\mathbf{x}(k)}. \quad (2.108)$$

The subscript "t" on $\mathbf{H}_t$ is used to emphasize that (2.108) is evaluated along the true trajectory $\mathbf{x}(k)$ rather than along the estimated trajectory $\hat{\mathbf{x}}(k)$ as in the usual EKF realization.

Performing the expectation of (2.105) in accordance with (2.101) and substituting (2.106), (2.107), and (2.108) yields Fisher's information matrix:

$$\begin{aligned} \mathbf{J}(N) &= \left(\frac{\partial \mathbf{x}(0)}{\partial \mathbf{x}(N)} \right)^T \mathbf{P}_0^{-1} \left(\frac{\partial \mathbf{x}(0)}{\partial \mathbf{x}(N)} \right) \\ &\quad + \sum_{k=0}^N \left(\frac{\partial \mathbf{x}(k)}{\partial \mathbf{x}(N)} \right)^T \mathbf{H}_t^T(k) \mathbf{R}^{-1}(k) \mathbf{H}_t(k) \left(\frac{\partial \mathbf{x}(k)}{\partial \mathbf{x}(N)} \right). \end{aligned} \quad (2.109)$$

Evaluation of the alternate representation for Fisher's information matrix in (2.102) is most easily accomplished through component-wise differentiation of (2.103). Consider a vector $\xi \in R^n$ and a scalar-valued function $\mathcal{F}(\xi)$ given by

$$\mathcal{F}(\xi) = \frac{1}{2} \xi^T \Gamma \xi, \quad (2.110)$$

where $\Gamma \in R^{n \times n}$ is positive definite symmetric. Equation (2.110) represents the general form of $\ln p_N$ in (2.103) which must be differentiated. Provided the components of ξ admit second derivatives with respect to components of a vector $\vartheta \in R^n$,

$$\begin{aligned} \left[\frac{\partial^2 \mathcal{F}(\xi)}{\partial \vartheta^2} \right]_{i,j} &= \frac{\partial^2 \mathcal{F}(\xi)}{\partial \vartheta_i \partial \vartheta_j} \\ &= \sum_{\ell,k} \left[\frac{\partial \xi_\ell}{\partial \vartheta_i} \Gamma_{\ell,k} \frac{\partial \xi_k}{\partial \vartheta_j} + \xi_\ell \Gamma_{\ell,k} \frac{\partial^2 \xi_k}{\partial \vartheta_i \partial \vartheta_j} \right] \\ &= \left[\left(\frac{\partial \xi}{\partial \vartheta} \right)^T \Gamma \left(\frac{\partial \xi}{\partial \vartheta} \right) \right]_{i,j} + \sum_{\ell,k} \left[\xi_\ell \Gamma_{\ell,k} \frac{\partial^2 \xi_k}{\partial \vartheta_i \partial \vartheta_j} \right]. \end{aligned} \quad (2.111)$$

Taking the expectation of (2.111) in accordance with (2.102) in a sequence of steps with $\vartheta = \mathbf{x}(N)$ and first with $\xi = (\mathbf{x}_0 - \bar{\mathbf{x}}_0)$ and then with $\xi = [\mathbf{z}(k) - \bar{\mathbf{z}}(k)]$, $k = 0, 1, 2, \dots, N$, which implies that in all steps ξ has zero mean and deterministic first and second partial derivatives, eliminates the second term on the right of (2.111) giving, with (2.106), (2.107), and (2.108), the previous result (2.109) as expected.

Finally, from (2.90)

$$\frac{\partial \mathbf{x}(k+1)}{\partial \mathbf{x}(N)} = \Phi_i(k) \frac{\partial \mathbf{x}(k)}{\partial \mathbf{x}(N)}, \quad \text{for } 0 \leq k \leq N-1, \quad (2.112)$$

where

$$\Phi_i(k) \triangleq \left. \frac{\partial \mathbf{f}[\mathbf{x}]}{\partial \mathbf{x}} \right|_{\mathbf{x}=\mathbf{x}(k)}, \quad \text{for } 0 \leq k \leq N-1 \quad (2.113)$$

is evaluated along the true trajectory as shown. In particular, with $\mathbf{I}_n$ representing the $n \times n$ identity matrix

$$\frac{\partial \mathbf{x}(N)}{\partial \mathbf{x}(N)} = \mathbf{I}_n = \left[\downarrow \prod_{i=N-1}^k \Phi_i(i) \right] \frac{\partial \mathbf{x}(k)}{\partial \mathbf{x}(N)}, \quad (2.114)$$

where the notation " $\downarrow \prod$ " indicates that the index i decreases from left to right in the matrix product. The Jacobian matrices in (2.109) are then computed as

$$\frac{\partial \mathbf{x}(k)}{\partial \mathbf{x}(N)} = \prod_{i=k}^{N-1} \Phi_i^{-1}(i). \quad (2.115)$$

The substitution of (2.115) into (2.109) gives a closed-form expression for Fisher's information matrix for the system and measurement models defined in (2.90) and (2.91):

$$\begin{aligned} \mathbf{J}(N) = & \left(\prod_{i=0}^{N-1} \Phi_i^{-1}(i) \right)^T \mathbf{P}_0^{-1} \left(\prod_{i=0}^{N-1} \Phi_i^{-1}(i) \right) \\ & + \sum_{k=0}^N \left(\prod_{i=k}^{N-1} \Phi_i^{-1}(i) \right)^T \mathbf{H}_i^T(k) \mathbf{R}^{-1}(k) \mathbf{H}_i(k) \left(\prod_{i=k}^{N-1} \Phi_i^{-1}(i) \right). \end{aligned} \quad (2.116)$$

A more useful form for practical computation is a recursive formulation which follows from (2.116) by inspection:

1. Initialization -

$$\mathbf{J}(0) = \mathbf{P}_0^{-1} + \mathbf{H}_0^T(0) \mathbf{R}^{-1}(0) \mathbf{H}_0(0); \quad (2.117)$$

2. Recursive evaluation for $k \geq 0$ -

$$\mathbf{J}(k+1) = [\Phi_i^{-1}(k)]^T \mathbf{J}(k) [\Phi_i^{-1}(k)] + \mathbf{H}_i^T(k) \mathbf{R}^{-1}(k) \mathbf{H}_i(k). \quad (2.118)$$

If the information matrix $\mathbf{J}(k)$ remains singular in the absence of prior information, i.e. $(\mathbf{P}_0^{-1} = 0)$, then one or more states are not observable. In general, however, observability of nonlinear systems is often difficult to establish.

The above derivation has been given for discrete-time, nonlinear system and measurement models. Furthermore, this derivation assumes an unbiased estimation procedure. A more general result for an estimator with state dependent bias $\mathbf{b}(\mathbf{x})$ is given by [55]

$$\mathbf{S}(N) \geq \left(\mathbf{I}_n + \frac{\partial \mathbf{b}(\mathbf{x})}{\partial \mathbf{x}} \right) \mathbf{J}^{-1} \left(\mathbf{I}_n + \frac{\partial \mathbf{b}(\mathbf{x})}{\partial \mathbf{x}} \right)^T, \quad (2.119)$$

where $\mathbf{S}(N)$ is defined in (2.99) and the Jacobian of $\mathbf{b}$ is evaluated along the true trajectory. However, the bias $\mathbf{b}(\mathbf{x})$ is generally not available in analytical form. Unfortunately, a state dependent bias can result in higher or lower bounds in comparison to $\mathbf{J}^{-1}$. For example, consider $\mathbf{b}(\mathbf{x}) = -\mathbf{x}/2$ which gives $\mathbf{S} \geq \mathbf{J}^{-1}/4$, while $\mathbf{b}(\mathbf{x}) = \mathbf{x}$ gives $\mathbf{S} \geq 4\mathbf{J}^{-1}$. As a result, most analysis proceed under the assumption of unbiased estimation resulting in approximate covariance bounds.

Broida and Chellappa [24, 26, 27, 31], Young and Chellappa [30], Taylor [44], and Chang [45] all employ Cramer-Rao analysis for filter performance evaluation. Broida and Chellappa, and Young and Chellappa do not consider *a priori* information. Taylor does consider information provided by the initial state estimate and treats the continuous-time system case. In the continuous-discrete filtering techniques of Section 2.3.3, results in [44] indicate that approximate covariance bounds can be computed as in (2.117) and (2.118), but with $\Phi_i(k)$ replaced with $\hat{\Phi}_i(k, k-1)$ defined in (2.78).

2.5 Chapter Summary

This chapter has introduced notation and reviewed fundamental theory required for filter development. The perspective projection image formation model of Section 2.2.1 is employed throughout the remainder of this work. In the reviews of Broida and Chellappa's methods, only the object-centred, F_O , and camera-centred, F_C , reference frames defined in Section 2.2.2 are required. However, the proposed technique for planar constrained motion also uses the earth-fixed reference system, F_E , which reduces the dimension of the estimation problem and leads to solutions that may be naturally extended to the case of a moving observer. All four representations of relative orientation (nine-parameter, Euler angle, exponential, and quaternion) introduced in Section 2.2.3 are required in the following development. Of the various filters discussed in Section 2.3, Broida and Chellappa's method of [24] is based on the discrete-time, extended Kalman filter with local iterations, while their method of [28] employs the continuous-discrete form of the filter with local iterations. The proposed method for planar motion uses the discrete-time, extended Kalman filter. Although satisfactory performance can be achieved in the case of known structure without local iterations, improved convergence rates are realized with the iterated filter for motion and structure estimation in the presence of feature point occlusion. The approximate Cramer-Rao bounds developed in Section 2.4 are used extensively in the following chapters to assess filter performance through comparison of these bounds with Monte Carlo simulation results.

Chapter 3

Planar Motion Viewed From The Plane

3.1 Introduction

Consider a single camera system observing an object whose motion is constrained to lie on a plane. If the plane of motion is known with respect to the camera system, and the optical axis pierces this plane in a single point, then the distance from the centre of projection (the origin of the camera reference frame, F_C) to the piercing point can be computed. In this case, depth information is contained in the analytical expression for the plane in F_C . These notions are made more precise in Chapter 4 which considers arbitrary viewing angles from points off of the plane. If, on the other hand, the optical axis lies in or is parallel to the plane of motion, knowledge of the plane in F_C does not yield depth information. This gives rise to a degenerate case of planar motion analysis and is the primary focus of this chapter.

Broida and Chellappa [24] have investigated this special case of the general motion analysis problem by considering the typical physical arrangement and image shown in Figure 3.1. In [24], the moving object, shown here as a block, is assumed to be undergoing constant translational velocity in a plane defined by the optical axis and central scan line of the imaging system. Constant rotational velocity is also assumed in which the axis of rotation is normal to the plane of translational motion. All feature points of interest, shown here as midpoints of vertical edges of the block, also lie on this plane and project onto the central scan line of the image. Only horizontal positions of feature points in the image are considered in this investigation.

Figure 3.1: Planar motion viewed from the plane. All feature points lie in the plane of motion which is defined by the optical axis and central scan line of the imaging system.

Broida and Chellappa refer to this case as one-dimensional (1D) "imagery" with 2D motion. In addition to providing a simplified problem for initial investigations, this case is also of practical significance in that the physical arrangement of Figure 3.1 may also arise in a large number of machine vision applications.

As discussed in Chapter 2, it is not possible to recover object motion and structure completely with a monocular imaging system in the absence of additional information. Moreover, knowledge of the plane of motion in this case does not yield essential depth information. In this chapter, some knowledge of object structure is assumed. The remaining structural parameters as well as object position, translational velocity, orientation, and rotational velocity are to be estimated given noisy measurements of feature positions in the image.

For some initial state estimate errors, the filters examined in this chapter had a tendency to converge reliably and steadily to motion and structural parameters *different* from those employed to generate the imagery. This phenomenon has also been noted by Broida [31] and Broida and Chellappa [24]. In [31] it is suggested that different sets of motion and structural parameters can lead to essentially the same image sequence, so failure to converge to the "true" parameters does not necessarily imply filter divergence in the usual sense. Instead, there appears to be at least two solutions in state space to which the filters may converge, particularly for the specialized geometry considered in this chapter. Uniqueness of motion and structural parameters, or the lack thereof, is of fundamental importance in the state estimation problem. For motion estimation problems in which only two image samples are considered, elegant proofs of uniqueness of motion parameters have been established for feature-based methods in [17]. Uniqueness of motion parameters has also been investigated for an arbitrarily long sequence of stereo images in [30] for constant acceleration and precession models. Under uniform time sampling, the motion and measurement models employed in [30] are shown to yield unique translational velocity and acceleration parameters, but angular velocity and precession parameters are said to be "never unique". A rigorous analytical treatment of uniqueness issues is beyond the scope of the thesis work. However, a demonstration of the ambiguity of rotational motion when observed through the perspective projection transforma-

tion will help to clarify the behavior of state estimation algorithms examined in this chapter.

Figure 3.2 shows a sequence of eight images for a rotating rectangular block with the viewing geometry of this chapter. There are two interpretations for the structure of the object in each image frame: the first is consistent with the fact that the object is a rectangular block (all corners 90°), while the second interpretation views the object as a skewed block with one end smaller than the other. The apparent motion of either interpretation can be followed from one frame to the next (down the left column and up the right) through all eight frames. The direction of rotation of the rectangular block (first interpretation) is counterclockwise referenced from above, whereas that associated with the skewed block (second interpretation) is perceived as clockwise referenced from above. This demonstration suggests that imagery generated from one trajectory in state space has two distinct interpretations. However, there must exist a true object and object motion which is consistent with the skewed-block interpretation. The question of uniqueness of parameters is concerned with whether or not multiple interpretations can be described with the parameterization of object motion and structure employed in the estimation algorithm, that is, whether or not multiple interpretations lie in the state space of the problem formulation.

Broida [31] has also observed convergence to alternate motion and structural parameters for their more general approach with transparent objects (no occlusion) which is investigated in Chapter 4. However, for the general viewing angles considered in Chapter 4, there were no cases in which either Broida and Chellappa's method or the proposed technique for planar motion failed to converge to within a reasonable interval of the true motion parameters. This problem will be discussed again in the introduction to Chapter 4. In all Monte Carlo simulations of the present chapter, convergence to alternate parameters was artificially detected, and sample errors associated with these runs were removed from Monte Carlo results.

Section 3.2 reviews the development of Broida and Chellappa's approach to this problem presented in [24]. Simulation analysis of their technique first show sample and Monte Carlo simulations for the trajectory employed in [24] in order to ver-

Figure 3.2: Two interpretations of rotational 2D motion. In each image frame, there are two interpretations of the object structure and the direction of rotational motion.

ify correct behavior of the filtering algorithm. This is followed by a comparison of Cramer-Rao bounds for several trajectories in order to demonstrate fundamental properties of their algorithm. In particular, these results confirm the intuitive notions that more information is available when the object lies on or near the optical axis, and that estimation accuracy can be expected to decrease as the object moves farther away from the camera. A small portion of the trajectory employed in [24] which, from the Cramer-Rao bound analysis, contains a relatively high level of information is selected for all further simulations of this chapter. The sample period is reduced from that employed in Broida and Chellappa's trajectory in order to provide a significant number of measurement events over the smaller trajectory and to demonstrate initial transients in the filter response.

In Section 3.3, a slightly alternate parameterization of object motion is proposed and evaluated assuming object structure is known completely. Further consideration of structure estimation is deferred to Chapter 4. The known structure case is considered in order to demonstrate filter performance in terms of the number of available feature points, and feature point occlusion. The number of feature points of interest determines the dimension of the measurement vector and hence the dimension of the matrix inversion problem in the Kalman filtering algorithm (see (2.57)). It is shown that increasing the number of feature points from two to three results in significant performance improvements, however, employing more than three feature points yields decreasing returns at the cost of increased computational complexity. Simulation results for motion estimation under feature point occlusion demonstrate that little if any degradation in performance results from the occlusion process, provided at least two feature points are available in each image.

The following sections share a common structure: the system geometry appropriate to the specific problem is described, followed by a development of system and measurement models, linearization of the models, and finally simulation results are presented. Within each section, simulation analysis consider distinct "Cases" associated with a particular set of trajectory, structure, and stochastic parameters. Each Case label includes a section number and run number in the form, "Case Section number-Run number". Graphical results shown for each Case may include a

representative image resulting from the particular trajectory, parameter estimation errors of a sample run, the approximate Cramer-Rao lower bounds (CRLB), and the root-mean-square-errors (RMSE) and mean errors (ME) of a 60 run Monte Carlo simulation. The ME and mean-square-errors (MSE) over 60 runs were computed as

$$\begin{aligned} \text{ME}(k) &= \frac{1}{60} \sum_{j=0}^{59} \tilde{\mathbf{x}}_j(k|k), \text{ and} \\ \text{MSE}(k) &= \frac{1}{60} \sum_{j=0}^{59} \text{diag}\{\tilde{\mathbf{x}}_j(k|k)\tilde{\mathbf{x}}_j^T(k|k)\}, \end{aligned} \quad (3.1)$$

where $\tilde{\mathbf{x}}_j(k|k)$ is the state estimation error, defined in (2.52), of the j th simulation run. The RMSE is taken as the square root of the elements of the vector MSE. Cramer-Rao bounds are computed from the true trajectory as shown in Chapter 2, Section 2.4.

3.2 Broida and Chellappa's Approach

This section reviews Broida and Chellappa's approach to the motion and structure estimation problem for the specialized geometry shown in Figure 3.1. It should be noted, however, that the geometry has been altered to reflect the axis orientation of the camera reference frame shown in Figure 2.1.

3.2.1 System Geometry

Figure 3.3 shows the assumed system geometry in which the earth-fixed reference frame F_E is aligned with the stationary camera-centered reference frame F_C . The 2D rigid body is modeled by its unknown centre of mass coordinates in F_E , (x, y) (here, $z(t) \equiv 0$), which translate with unknown constant velocity $(\dot{x}, \dot{y})$. Two feature points, displaced from the mass centre by moment arm lengths r_1 and r_2 , have unknown phase angles p_1 and p_2 , respectively, which rotate about the mass centre with constant angular velocity $\mathbf{w} \equiv [0, 0, w]^T$. It is assumed that r_1 , which accounts for a global scale factor associated with the estimates of the position and velocity of the mass centre, is known or estimated as is the ratio r_2/r_1 . If additional

Figure 3.3: System geometry for Broida and Chellappa's 2D approach.

feature points are available, their phase angles and moment arm lengths are added in a straight forward way to the model while their moment arm lengths are assumed known in terms of r_1 . The measurement model is assumed to be given by perspective projection as described in Section 2.2.1 with the optical centre at the origin, the optical axis aligned with the positive x_C -axis, y_C parallel to image scan lines, and the focal length being represented by L . Broida and Chellappa employ a transparent object assumption so that feature occlusion need not be considered.

3.2.2 Motion and Structural Model

With only two feature points available, the state vector can be written as

$$\mathbf{x}(t) = [x, \dot{x}, y, \dot{y}, p_1, p_2, w]^T. \quad (3.2)$$

A noiseless state model is employed to describe unforced motion of the object as

$$\dot{\mathbf{x}}(t) = [\dot{x}, 0, \dot{y}, 0, w, w, 0]^T. \quad (3.3)$$

With a possibly time varying sample period T_k , (3.3) can be written as a discrete-time difference equation

$$\mathbf{x}(k+1) = \Phi(k)\mathbf{x}(k), \quad (3.4)$$

where

$$\mathbf{x}(k) = \begin{bmatrix} x(k) \\ \dot{x}(k) \\ y(k) \\ \dot{y}(k) \\ p_1(k) \\ p_2(k) \\ w(k) \end{bmatrix}, \text{ and } \Phi(k) = \begin{bmatrix} 1 & T_k & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & T_k & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & T_k \\ 0 & 0 & 0 & 0 & 0 & 1 & T_k \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \quad (3.5)$$

Equation (3.4) is linear in the elements of the state vector of (3.2). Dynamic equations which describe general 3D, six-degree-of-freedom motion remain linear in translational states, but time propagation of rotational motion is nonlinear. With

a linear system model, even though the measurement model will be nonlinear, the time update of the Kalman state estimate between measurements follows (2.56). This stage of the filter is further simplified in this particular case since the state transition matrix $\Phi(k)$ is relatively sparse and will be constant under time invariant sampling. As discussed in Section 2.3.1, a noiseless system model assumes that object dynamics are known exactly which means the filter will eventually neglect further measurements as the gains decrease over time. The initial performance of the Kalman filter for this case, therefore, is largely dependent on the object trajectory, the nonlinear measurement model, and measurement noise levels.

3.2.3 Measurement Model

Following the discussion of Section 2.2, the horizontal image positions, u^1 and u^2 , of the two feature points form the measurement model through central projection geometry as

$$\mathbf{z}(k) = \begin{bmatrix} u^1(k) \\ u^2(k) \end{bmatrix} = \mathbf{h}[\mathbf{x}(k)] + \mathbf{v}(k), \quad (3.6)$$

where

$$\mathbf{h}[\mathbf{x}(k)] = \begin{bmatrix} h_1(k) \\ h_2(k) \end{bmatrix} = \begin{bmatrix} L(y + r_1 \sin p_1)/(x + r_1 \cos p_1) \\ L(y + r_2 \sin p_2)/(x + r_2 \cos p_2) \end{bmatrix}. \quad (3.7)$$

and L is the focal length. The measurement noise sequence $\mathbf{v}(k)$ is assumed to be zero-mean Gaussian, spatially correlated, and temporally white:

$$\mathcal{E} \{ \mathbf{v}(k) \mathbf{v}^T(j) \} = R(k) \delta_{kj}, \quad (3.8)$$

where $R(k)$ is the known spatial correlation matrix.

The Jacobian of $\mathbf{h}[\mathbf{x}]$, which is required in the EKF measurement update equations (2.68), is given by the 2×7 matrix

$$\mathbf{H}(k) = \frac{\partial \mathbf{h}}{\partial \mathbf{x}} = [H_{ij}], \quad (3.9)$$

where, in this case,

$$\begin{aligned}
 H_{i1} &= \frac{\partial h_i}{\partial x} = -h_i/(x + r_i \cos p_i), \quad i = 1, 2 \\
 H_{i3} &= \frac{\partial h_i}{\partial y} = L/(x + r_i \cos p_i), \quad i = 1, 2 \\
 H_{15} &= \frac{\partial h_1}{\partial p_1} = -L \frac{r_1^2 + x r_1 \cos p_1 + y r_1 \sin p_1}{(x + r_1 \cos p_1)^2}, \text{ and} \\
 H_{26} &= \frac{\partial h_2}{\partial p_2} = -L \frac{r_2^2 + x r_2 \cos p_2 + y r_2 \sin p_2}{(x + r_2 \cos p_2)^2},
 \end{aligned} \tag{3.10}$$

while all other entries of $\mathbf{H}$ are zero.

With two feature points, each image frame provides two equations in seven unknowns. At least four image frames are required, therefore, to obtain more equations than unknowns. In [24], a descriptive measure of the degree of smoothing, or redundancy, that can be achieved is taken as the ratio of the number of equations to the number of unknowns. With the model discussed here, redundancy increases linearly with the number of image frames. Each additional feature point adds one more unknown state variable to the dynamic model and one more equation to the measurement model.

3.2.4 Results and Performance With Simulated Imagery

The geometry of Figure 3.3 was used to generate sequences of image feature points according to (3.6). The focal length was set to unity for this analysis. The translating, rotating, transparent object was defined by its two feature points with moment arm lengths $r_1 = r_2 = 5$ units which enclose a constant angle of $p_1(t) - p_2(t) = 2$ radians. These parameters give the object a maximum physical size of approximately 8.5 units. In all cases considered here, a pseudo random Gaussian noise sequence with standard deviation $\sigma_v = 0.01$ units was added to true feature point positions in the image. These parameters are identical to those used by Broida and Chellappa in [24].

When the object is displaced in the direction of the optical axis by at least 50 units from the focal point, the maximum size of the object in the image is approx-

imately 0.2 units. The additive noise of standard deviation 0.01 units, therefore, represents quantities in excess of 5% of the object image size. A thorough investigation of filter performance under various measurement noise levels has been presented in [31]. Results of that investigation are fairly accurately predicted by Cramer-Rao bound analysis which, from (2.116) with measurement noise covariance $\mathbf{R}(k) \equiv \sigma_v^2 \mathbf{I}_m$ and constant sampling period T , employs Fisher's information matrix for this case as

$$\mathbf{J}(N) = (\Phi^{-N})^T \mathbf{P}_0^{-1} (\Phi^{-N}) + \frac{1}{\sigma_v^2} \sum_{k=0}^N (\Phi^{-(N-k)})^T \mathbf{H}_t^T(k) \mathbf{H}_t(k) (\Phi^{-(N-k)}), \quad (3.11)$$

where Φ is a constant matrix and $\mathbf{H}_t(k)$ is evaluated along the true trajectory. In the absence of prior state information, $\mathbf{P}_0^{-1} = \mathbf{0}$, inversion of $\mathbf{J}(N)$ gives approximate estimation error covariance lower bounds which vary directly with measurement noise levels. All simulation studies of this chapter fix σ_v at the modest level of 0.01 units.

The position of the object $(x(t), y(t))$ has been defined as a point which lies on the plane of motion at the prescribed distances (r_i) from feature points and moves with constant translational velocity. In the case of unforced motion, provided the object is undergoing rotational motion, the line parallel to the rotational axis and passing through the centre-of-gravity of the object intersects the plane of motion at the point $(x(t), y(t))$. If, on the other hand, object motion is purely translational, all points on or in the object which also lie on the plane of motion are equally likely candidates for $(x(t), y(t))$. For a trajectory with zero rotational rates, therefore, the system developed above is not observable. However, object motion and structure may still be recovered in the absence of rotational motion through the use of a reduced motion model which removes rotational states and which defines $(x(t), y(t))$ to lie at a specified feature point. Hence, in addition to all assumptions employed above, this technique also requires *a priori* knowledge concerning the presence or absence of rotational motion. A major consequence of this property of the estimation algorithm is serious performance degradation as rotational rates decrease.

As indicated in [24], the extended Kalman filter does not give satisfactory performance for this particular estimation problem. An iterated extended Kalman filter is employed with time update taken from the linear filtering equations in (2.56), measurement update equations taken from the extended filtering equations in (2.68), and local iterations from (2.86)-(2.89). A maximum of 20 iterations per measurement is imposed with termination if the maximum element of the difference in consecutive iterates is less than 0.01. Usually, 3 or 4 iterations per measurement were sufficient, except for the first few image frames in each run in which 15 to 20 iterations were common.

Simulation studies for this estimation technique include the following three cases:

Case 3.2-1 Broida and Chellappa's simulation in [24] is reproduced, to the extent possible, primarily to verify correct behavior of the filtering algorithm and to set the context for further discussions. Monte Carlo simulation results are also shown and compared to Cramer-Rao bounds.

Case 3.2-2 A comparison of Cramer-Rao bounds is given for several trajectories in order to demonstrate fundamental properties of the estimation algorithm. These results show that more information is available when the object lies on or near the optical axis, and that estimation accuracy decreases as the object moves farther away from the camera. Cramer-Rao bounds are also shown for decreasing angular rates for which filter performance is seriously degraded.

Case 3.2-3 Sample and Monte Carlo simulation results are shown for a small portion of the trajectory employed in [24] which, from the Cramer-Rao bound analysis, contains a relatively high level of information. The sample period is reduced from that employed in Case 3.2-1 in order to provide a significant number of measurement events over the smaller trajectory. Even with the larger number of measurements, initial transients in the filter response are significantly larger than those for Case 3.2-1. All further simulations of this chapter employ this trajectory.

This investigation concludes with a summary of results and a general discussion of Broida and Chellappa's technique.

Table 3.1: Simulation parameters for Case 3.2-1

$\mathbf{x}$	$\mathbf{x}(0)$	$\hat{\mathbf{x}}_0$	$\sigma(\hat{\mathbf{x}}_0)$	$T = 1 \text{ Sec.}$
x	50.0	52.0	5.0	units
$\dot{x}$	0.5	0.0	3.0	units/s
y	-30.0	-24.0	5.0	units
$\dot{y}$	1.0	0.0	3.0	units/s
p_1	0.5	1.5	0.5	radians
p_2	-1.5	0.0	0.5	radians
w	-1.0	-0.5	0.25	radians/s

Case 3.2-1: Broida and Chellappa's trajectory. The true object trajectory used in this simulation is identical to that for which results are shown in [24] with their constant sampling period of $T = 1.0$ second. Table 3.1 lists the simulation parameters for this case: the true initial state $\mathbf{x}(0)$, a sample initialization $\hat{\mathbf{x}}_0$, and standard deviations of initialization errors $\sigma(\hat{\mathbf{x}}_0)$ employed in a Monte Carlo simulation. Figure 3.4 demonstrates the true feature image points and a sample noisy image for this case over 60 seconds. In these plots, time is on the vertical axis and the "image" of the object (the distance between feature points in the image plane) at any time instant is the horizontal distance between the solid and dashed curves. This trajectory represents object motion with relatively high rotational velocity. Over the 60 second interval, the object translates 60 units parallel to the image plane (line) and 30 units away from the camera, and completes about 9.5 revolutions. Moreover, measurement noise levels are quite small in comparison to feature point displacement in the image between samples.

Figure 3.5 shows sample results for this trajectory when the filter was initialized as in [24] with $\hat{\mathbf{x}}(0| - 1) = \hat{\mathbf{x}}_0$ given in Table 3.1. The initial covariance matrix (not available in [24]) was taken as $\hat{\mathbf{P}}(0| - 1) = \mathbf{P}_0$, where

$$\mathbf{P}_0 = \text{diag}[\sigma^2(\hat{\mathbf{x}}_0)], \quad (3.12)$$

with $\sigma(\hat{\mathbf{x}}_0)$ as shown in Table 3.1. In this case, some initial transients are seen in the centre-of-gravity (CG) position errors over the first 5 seconds followed by

gradual convergence to within a reasonable interval of the true position by 30 seconds. Velocity errors were somewhat erratic during the first 10 seconds after which good convergence is shown. Estimates of angular quantities and the rotational rate converge after about 10 seconds.

A Monte Carlo simulation consisting of 60 runs was conducted with random filter initialization taken in accordance with (2.92) as

$$\hat{\mathbf{x}}(0) - \mathbf{x}(0) = \boldsymbol{\eta}, \quad (3.13)$$

where $\boldsymbol{\eta} \sim N(0, \mathbf{P}_0)$, and $\mathbf{P}_0$ is defined in (3.12). In this case, modest initialization errors are attributed to initial position, structure (p_1 and p_2), and angular rates, whereas translational velocity initialization errors are large in comparison to truth values. The root-mean-square-error (RMSE), Cramer-Rao lower bounds (CRLB), and sample mean errors (ME) over the 60 second trajectory are shown in Figure 3.6. As discussed in the introduction of this chapter, the filter sometimes failed to converge to the "true" motion parameters, but instead appeared to converge in some states to an incorrect value. Usually, this alternate form of convergence in some states resulted in an estimated trajectory with the opposite rotational velocity (negative of the true rate). All runs for which the filter converged to motion parameters other than those employed to generate the imagery were artificially detected and removed from the Monte Carlo simulation. In this particular case, the filter converged to alternate results during 4 runs, so the simulation was actually conducted over 64 runs with these 4 runs removed. Filter performance without considering these 4 runs is very good in that RMS errors over the 60 second interval are very close to Cramer-Rao bounds. In addition, the filter bias which is indicated by the ME trace over these runs is within reasonable limits after 15 seconds. The peaks seen in the RMSE and CRLB traces in x - and y -position errors and in p_2 -angular position errors over the first 20 seconds appear to occur when the image of the object has zero width.

Figure 3.4: Image plane positions (units) of feature points for Case 3.2-1.

Figure 3.5: Sample results for Case 3.2-1.

Figure 3.6: Monte Carlo simulation results for Case 3.2-1. All plots show — RMSE, --- CRLB, and ME.

Case 3.2-2: Cramer-Rao bound analysis for various object trajectories. This case demonstrates the behavior of approximate covariance bounds for a very "wide" trajectory (translational motion parallel to the image plane), a very "deep" trajectory (translational motion parallel to the optical axis), and for trajectories which remains close to the optical axis at a reasonable depth, but have decreasing angular rates.

In Figure 3.6 above, the Cramer-Rao bound for y -position (lateral) estimation error increases after the object crosses the optical axis at 30 seconds, while the bound for x -position (depth) estimation error appears to remain almost constant after 30 seconds. Figure 3.7 shows the behavior of the Cramer-Rao bounds in the absence of prior information ($\mathbf{P}_0^{-1} = 0$) for a very wide trajectory of constant depth defined by the initial state

$$\mathbf{x}(0) = \begin{bmatrix} x(0) \\ \dot{x}(0) \\ y(0) \\ \dot{y}(0) \\ p_1(0) \\ p_2(0) \\ w(0) \end{bmatrix} = \begin{bmatrix} 50.0 \text{ units} \\ 0.0 \text{ units/s} \\ -100.0 \text{ units} \\ 1.0 \text{ units/s} \\ 0.5 \text{ radians} \\ -1.5 \text{ radians} \\ -1.0 \text{ radians/s} \end{bmatrix}, \quad (3.14)$$

with unity sample period over a 200 second time interval. The CRLB for y -position estimation is lowest as the object crosses the optical axis, and then increase as the object moves to the right. As the object moves away from the optical axis, angles between rays joining the centre of projection to feature points decrease and approach zero in the limit. This results in increased linear dependence in the system of equations relating the state variables to image feature positions. Of course in practical applications, finite dimensions of the image plane must be considered. However, these results suggest that degraded filter performance should be expected when the observed object trajectory lies near the boundaries of the image, whereas active control of the camera to maintain the object near the centre of the image may lead to improved state estimation.

Figure 3.8 demonstrates the behavior of the Cramer-Rao bounds (again with $\mathbf{P}_0^{-1} = 0$) for a very deep trajectory on the optical axis defined by the initial state

$$\mathbf{x}(0) = \begin{bmatrix} x(0) \\ \dot{x}(0) \\ y(0) \\ \dot{y}(0) \\ p_1(0) \\ p_2(0) \\ w(0) \end{bmatrix} = \begin{bmatrix} 30.0 \text{ units} \\ 0.5 \text{ units/s} \\ 0.0 \text{ units} \\ 0.0 \text{ units/s} \\ 0.5 \text{ radians} \\ -1.5 \text{ radians} \\ -1.0 \text{ radians/s} \end{bmatrix}, \quad (3.15)$$

with unity sample period over a 200 second time interval. In this case, the CRLB for x -position estimation as well as angular position increase as the object moves away from the camera. As in the wide trajectory case, rays joining the centre of projection to feature points become more parallel as depth increases and hence reduced estimation accuracy can be expected. This suggests that active control of focal length to maintain significant size of the object image may lead to improved performance.

As previously discussed, this system is not observable for zero rotational velocity. With $w = 0$, Fisher's information matrix remains poorly conditioned for inversion regardless of the number of image frames considered. Broida and Chellappa's trajectory of Case 3.2-1 had fairly high rotational rates in comparison to translational modes. It is interesting to note that only relatively high rotational rates are considered in simulations and experiments with real imagery in [31]. Figure 3.9 shows Cramer-Rao bounds for trajectories with an initial state

$$\mathbf{x}(0) = \begin{bmatrix} x(0) \\ \dot{x}(0) \\ y(0) \\ \dot{y}(0) \\ p_1(0) \\ p_2(0) \\ w(0) \end{bmatrix} = \begin{bmatrix} 50.0 \text{ units} \\ 0.5 \text{ units/s} \\ -5.0 \text{ units} \\ 1.0 \text{ units/s} \\ 0.5 \text{ radians} \\ -1.5 \text{ radians} \\ w_0 \text{ radians/s} \end{bmatrix}, \quad (3.16)$$

were $\omega_0 = 1.0, 0.5, 0.2$, and 0.1 radians per second. The sampling period in this case is taken as $T = 1/30$ seconds through a total time of 10 seconds. Cramer-Rao bounds for all states increase as angular velocity decreases. In particular, bounds on the CG position and angular position errors appear to be the most sensitive to decreasing angular velocity.

Figure 3.7: Cramer-Rao bounds for a very wide trajectory.

Figure 3.8: Cramer-Rao bounds for a very deep trajectory.

Figure 3.9: Cramer-Rao bound variation with angular velocity. All plots show — $w = 1.0$, - - - $w = 0.5$, $w = 0.2$, and - . - . - $w = 0.1$ radians/s.

Table 3.2: Simulation parameters for Case 3.2-3

x	$x(0)$	$\hat{x}_0$	$\sigma(\hat{x}_0)$	$T = 1/30$ Sec.
x	50.0	55.0	5.0	units
$\dot{x}$	0.5	0.0	3.0	units/s
y	-5.0	-2.0	5.0	units
$\dot{y}$	1.0	0.0	3.0	units/s
p_1	0.5	1.5	0.5	radians
p_2	-1.5	0.0	0.5	radians
w	-1.0	-0.5	0.25	radians/s

Case 3.2-3: Central trajectory, high rotational rates, and low translational rates. Table 3.2 lists the true initial state $x(0)$, a sample filter initialization, $\hat{x}_0$, and the assumed standard deviation of initialization errors, $\sigma(\hat{x}_0)$. In this case, the object trajectory remains close to the optical axis, translates from 50 to 55 units in depth, and still has fairly high rotational rates of 1 radian/s. The sample period in this case is taken as $T = 1/30$ seconds. Results of the Cramer-Rao analysis of Case 3.2-2 indicate that performance bounds are relatively low for this trajectory. However, the following results demonstrate that covariance bounds do not predict filter performance over the beginning of the run as accurately as they did in Case 3.2-1.

Figure 3.10 shows true and sample images for this trajectory. The object translates 10 units parallel to the image plane and only 5 units in depth, while rotating through 573° over the 10 second interval. As in Case 3.2-1, the standard deviation of measurement noise was 0.01 units, however comparison with Figure 3.4 indicates that measurement noise is far more apparent in this case due to the smaller inter-sample motion. Figure 3.11 shows results from a sample run initialized with $\hat{x}_0$ as shown in Table 3.2. The x -position (depth) and velocity errors are degraded in this case, not only during initial transients, but seem to remain biased throughout the run. The y -position (lateral) and velocity errors, as well as angular position and rate errors have converged in approximately 3 seconds. Figure 3.12 shows results of the

Monte Carlo simulation for this case. This simulation required 104 runs over the 10 second image sequence in which 44 runs converged to alternate sets of motion and structural parameters. In contrast to Case 3.2-1, the x - and y -position RMS errors increase significantly above the Cramer-Rao bounds during the initial portion of the trajectory, followed by a rapid decrease after 3 seconds. The x -velocity errors show significant bias over the first 3 seconds and remain above the Cramer-Rao bounds throughout the trajectory (recall that the true x -velocity is 0.5 units/s). The y -velocity errors demonstrate good convergence as do the angular position and rate errors. The angular position RMS error of p_2 increases over the short time interval about 3 seconds during which the x - and y -position RMS errors show a rapid decrease.

Figure 3.10: Image plane positions (units) of feature points for Case 3.2-3.

Figure 3.11: Sample results for Case 3.2-3.

Figure 3.12: Monte Carlo simulation results for Case 3.2-3. All plots show — RMSE, --- CRLB, and ME.

3.2.5 Discussion

Broida and Chellappa's approach to this specialized problem provides an introduction to their more general approach of [28, 31]. In this 2D motion case with specific viewing angles, some or all structure must be known *a priori* in order to recover translational parameters. To assume that moment arm lengths to each feature point are known, which is the case here, means essentially that the object itself is known and has been physically measured. In this case it is unlikely that moment arm angles would remain totally unknown. Moreover, even if an application demands that initial estimates of these angles are to be refined through recursive estimation, the motion model employed here has not fully exploited the rigid object assumption. Structural parameters of a rigid object can be represented with respect to an object-centred reference frame by constant quantities. For example, in the geometry of Figure 3.3, $p_1 - p_2$ is a constant value. A dynamic model which includes these constants as states may be decoupled to eliminate further estimation of constant structural parameters as the confidence in these parameter values reaches some predefined threshold. This practice would not only provide a means to reduce the dimension of the estimation problem, and therefore the computational complexity, over time, but in addition emphasis could be shifted to structural parameter estimation for new features in order to construct a more meaningful and complete representation of the object being observed. Indeed, the work of Ayache and Faugeras [34] provides an excellent example of this practice.

Generally, the simulation results for Broida and Chellappa's approach demonstrate adequate filter performance, provided object rotational rates are relatively large (Case 3.2-1). Degraded performance as rotational rates decrease is a significant disadvantage of this approach to motion analysis. Even with the larger number of measurements available through reduction of the sample rate, and the high level of information available in the trajectory of Case 3.2-3, filter performance is degraded particularly during transients over the initial portion of the trajectory. Good convergence in most states is shown once a significant level of information has been accumulated in the filtering process. Poor performance during initial transients

may have contributed to the larger number of runs which converged to alternate results (44 for Case 3.2-3 compared to 4 for Case 3.2-1). Of all states, the filter seemed to have the most difficulty estimating x -position (depth) and velocity of the object, however, this is not really surprising given the loss of depth information in the perspective projection image formation model.

3.3 Motion Estimation Under Known Structure

A primary observation of Section 3.2 was that Broida and Chellappa's technique does not fully exploit the rigid object assumption. In this section, an alternate parameterization for object motion is proposed in the context of this specialized problem. The motion model developed here is directly applicable to the case of general planar motion with arbitrary viewing angles considered further in Chapter 4. The measurement model in this chapter, however, is of reduced dimension since only horizontal image feature positions are considered. In addition, the geometry of this specialized problem aligns the camera reference system with the earth-fixed frame.

This investigation focuses on motion parameter estimation when the structure of the object is known. Object structure is defined by a set of known constant feature points, $\mathcal{S} = \{\mathbf{r}_O^i | i = 1, 2, \dots, N_p\}$. Each $\mathbf{r}_O^i$ represents a vector describing the location of the i th feature point, p^i , with respect to an object-centred reference frame as described in Section 2.2.2. In practice the known structure assumption implies that the object of interest has already been recognized and associated with the set $\mathcal{S}$. Furthermore, the nature of the correspondence problem is altered in the sense that features detected in the image must, at least initially, be related to known feature points on the object. This correspondence between detected features and elements of the set $\mathcal{S}$ may in fact be more difficult to establish than the correspondence of detected features between image frames. However, the known structure case will provide a mechanism under which filter performance can be assessed in the estimation of motion parameters alone.

3.3.1 System Geometry

Figure 3.13 shows the system geometry that will be used in this investigation. As described above, the object is defined by its feature points, p^i , which are expressed with respect to an object-centred reference frame, F_O , through the vectors $\mathbf{r}_{O'}^i$. The origin of F_O , denoted O_O , is not observed directly, but lies on the axis of rotation of the object. That is, this point defines the position of the rotational axis which is normal to the plane of motion. As in the previous section, the stationary camera-centred reference frame, F_C , is aligned with the earth-fixed frame, F_E . Translational motion of the object is modeled by the position of O_O , $(x(t), y(t))$, and (assumed) constant velocity $(\dot{x}, \dot{y})$ in F_C . The orientation of F_O relative to F_C is described by the yaw angle $\psi(t)$ and rotational velocity $\mathbf{w} \equiv [0, 0, w]^T$. The measurement model is assumed to be given by perspective projection as in Section 2.2.1 with the optical centre at the origin, the optical axis aligned with the positive x_C -axis, y_C parallel to image scan lines, and with focal length L .

3.3.2 Motion Model

The state vector in this case is

$$\mathbf{x}(t) = [x, \dot{x}, y, \dot{y}, \psi, w]^T. \quad (3.17)$$

Again, a noiseless state model is employed to describe unforced motion of the object as

$$\dot{\mathbf{x}}(t) = [\dot{x}, 0, \dot{y}, 0, w, 0]^T. \quad (3.18)$$

Assuming a sample period T_k , (3.18) can be written in discrete-time form as

$$\mathbf{x}(k+1) = \Phi(k)\mathbf{x}(k), \quad (3.19)$$

Figure 3.13: System geometry for motion parameter estimation.

where

$$\mathbf{x}(k) = \begin{bmatrix} x(k) \\ \dot{x}(k) \\ y(k) \\ \dot{y}(k) \\ \psi(k) \\ \dot{\psi}(k) \end{bmatrix}, \text{ and } \Phi(k) = \begin{bmatrix} 1 & T_k & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & T_k & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & T_k \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \quad (3.20)$$

Equation (3.19) is linear in the elements of the state vector of (3.17). The dimension of the state vector is independent of the number of match points due to the known structure assumption. This model also assumes constant translational and rotational velocity. However, under a planar motion assumption, higher translational and rotational time derivatives can be incorporated into the model which remains linear in the state vector.

3.3.3 Measurement Model

The position of object feature points can be expressed with respect to F_C , as described in Section 2.2.2, through the rotation matrix

$$\mathbf{I}_C^O = \begin{bmatrix} \cos \psi & -\sin \psi \\ \sin \psi & \cos \psi \end{bmatrix}, \quad (3.21)$$

and translation vector

$$[\mathbf{T}_O^C]_C = \begin{bmatrix} x \\ y \end{bmatrix}, \quad (3.22)$$

according to the relationship

$$\mathbf{r}_C^i = \mathbf{I}_C^O \mathbf{r}_O^i + [\mathbf{T}_O^C]_C. \quad (3.23)$$

Equation (3.23) can be written in component form as in the first two equations of (2.9). The horizontal image position, $h^i[\mathbf{x}(k); \mathbf{r}_O^i]$ of the i th feature point is then given by the perspective projection transformation of Equation (2.2),

$$h^i[\mathbf{x}(k); \mathbf{r}_O^i] = \frac{r_{C,2}^i}{r_{C,1}^i} = L \frac{[0, 1] (\mathbf{I}_C^O \mathbf{r}_O^i + [\mathbf{T}_O^C]_C)}{[1, 0] (\mathbf{I}_C^O \mathbf{r}_O^i + [\mathbf{T}_O^C]_C)}, \quad (3.24)$$

where L is the focal length. With N_p feature points of interest, and initially assuming a transparent object, the measurement model is given by

$$\mathbf{z}(k) = \begin{bmatrix} u_1(k) \\ u_2(k) \\ \vdots \\ u_{N_p}(k) \end{bmatrix} = \mathbf{h}[\mathbf{x}(k); \mathcal{S}] + \mathbf{v}(k), \quad (3.25)$$

where

$$\mathbf{h}[\mathbf{x}(k); \mathcal{S}] = \begin{bmatrix} h^1[\mathbf{x}(k); \mathbf{r}_O^1] \\ h^2[\mathbf{x}(k); \mathbf{r}_O^2] \\ \vdots \\ h^{N_p}[\mathbf{x}(k); \mathbf{r}_O^{N_p}] \end{bmatrix}. \quad (3.26)$$

The Jacobian of $\mathbf{h}[\mathbf{x}; \mathcal{S}]$, which is required in the EKF realization, is given by the $N_p \times 6$ matrix

$$\mathbf{H}(k) = \frac{\partial \mathbf{h}}{\partial \mathbf{x}} = [H_{ij}], \quad (3.27)$$

where, in this case,

$$\begin{aligned} H_{i1} &= \frac{\partial h^i}{\partial x} = -h^i / r_{C,1}^i \\ H_{i3} &= \frac{\partial h^i}{\partial y} = L / r_{C,1}^i \\ H_{i5} &= \frac{\partial h^i}{\partial \psi} = -L \frac{(\mathbf{r}_C^i)^T \mathbf{I}_C^O \mathbf{r}_O^i}{(r_{C,1}^i)^2} \end{aligned} \quad (3.28)$$

while all other entries of $\mathbf{H}$ are zero. A single image provides N_p equations in 6 unknown state variables. Each additional feature point provides one more equation, but does not increase the dimension of the state vector.

3.3.4 Feature Point Occlusion

As the object rotates and translates, feature points naturally become occluded and may reappear at some future time depending on the object trajectory. Another form of occlusion occurs when the object becomes partially or completely hidden from view as it moves behind another object in the foreground of the scene. These two

forms of occlusion can be treated identically in the state estimation stage. With reference to Figure 3.1 the occlusion problem due to motion of a solid block for the specialized viewing angles considered here is demonstrated by the dashed lines and dashed circles denoting edges and features which are not seen in the image. In any given image, at most three, but at least two of the four known feature points are visible. In this case, the dimension of the measurement vector $\mathbf{z}(k)$ is time varying, i.e., $\mathbf{z}(k) \in R^{m(k)}$, where here $m(k) = N_v(k)$ and $N_v(k)$ is the number of visible feature points in the k th image. The arrival of each sampled image requires first that a correspondence between detected features and a subset $\mathcal{S}_v(k) \subseteq \mathcal{S}$ of structural parameters be established. The innovation in the measurement update stage of each iteration of the filter is computed as

$$\eta(k) = \mathbf{z}(k) - \mathbf{h}[\hat{\mathbf{x}}(k|k-1); \mathcal{S}_v], \quad (3.29)$$

where $\mathbf{h}[\hat{\mathbf{x}}(k|k-1); \mathcal{S}_v]$ is defined as in (3.26) and $\hat{\mathbf{x}}(k|k-1)$ is the Kalman state prediction defined in (2.68). The Jacobian matrix of (3.27) is computed as in (3.28) where the index i now refers to the i th feature position measurement and corresponding element of $\mathcal{S}_v$, rather than to elements of $\mathcal{S}$.

3.3.5 Results and Performance With Simulated Imagery

The geometry of Figure 3.13 was used to generate sequences of image feature points with correspondence information. In all cases, the focal length was set to unity, and the sample period is taken as $T = 1/30$ seconds. Independent Gaussian noise with standard deviation 0.01 units was added to the location of each feature point in the image. For the objects considered in the following simulations, this measurement noise level represents quantities in excess of 5% of the object image size. As in Broida and Chellappa's approach of the previous section, an iterated extended Kalman filter is employed with a maximum of 20 iterations per measurement and terminating if the maximum element of the difference of consecutive iterates is less than 0.01. Usually, 2 or 3 iterations per measurement were sufficient in this case, except for the first few image frames in each run in which 12 to 20 iterations were common.

Simulation results for this estimation technique include the following three cases:

Case 3.3-1 Three feature points without occlusion are employed in a trajectory identical to Case 3.2-3. As demonstrated in the Cramer-Rao bound analysis of Case 3.2-2, these motion parameters are representative of those for which covariance bounds are relatively low. Sample and Monte Carlo simulation results are shown, but here angular position and rate initialization errors are larger than those of Case 3.2-3. These results show significant improvements over those for Case 3.2-3 due to the inclusion of one additional feature point.

Case 3.3-2 Sample and Monte Carlo simulation results are shown for four feature points with occlusion due to object motion. Results of this case suggest that little if any degradation in filter performance should be expected in the presence of feature occlusion provided at least two feature points are visible at all times.

Case 3.3-3 A comparison of Cramer-Rao bounds is given for two, three, and four feature points without occlusion, and for four feature points with occlusion. These results show that increasing the number of feature points from two to three results in significant decrease in performance bounds, however, employing more than three feature points yields decreasing returns at the cost of increased computational complexity.

This investigation concludes with a summary of results, and a general discussion of this technique in comparison to Broida and Chellappa's approach.

Case 3.3-1: Central trajectory, low translation, high rotation, and three feature points without occlusion. Initialization parameters for this case are shown in Table 3.3. This simulation employs the same trajectory as Case 3.2-3, but with larger initialization errors for angular position and velocity. The object trajectory again remains close to the optical axis at 50 to 55 units in depth and has a fairly high rotational rate of 1 radian/s. The sample period was taken as $T = 1/30$ seconds. Three feature points are now assumed to be known which define the structure set

$$\mathcal{S} = \left\{ \begin{bmatrix} 0.0 \\ 7.0 \end{bmatrix}, \begin{bmatrix} 7.0 \\ 0.0 \end{bmatrix}, \begin{bmatrix} -7.0 \\ -7.0 \end{bmatrix} \right\}. \quad (3.30)$$

Table 3.3: Simulation parameters for Case 3.3-1

$\mathbf{x}$	$\mathbf{x}(0)$	$\dot{\mathbf{x}}_0$	$\sigma(\dot{\mathbf{x}}_0)$	$T = 1/30$ Sec.
x	50.0	55.0	5.0	units
$\dot{x}$	0.5	0.0	3.0	units/s
y	-5.0	-3.0	5.0	units
$\dot{y}$	1.0	0.0	3.0	units/s
ψ	-1.0	0.0	1.0	radians
w	-1.0	0.0	1.0	radians/s

Figure 3.14 shows true and sample images for this trajectory. A comparison with Figure 3.10 indicates that the third feature point appears to provide a significant level of addition information about the motion of the block. The overall size of the object in the image is never zero since the feature points are non-collinear, and the apparent centroid (mean location) of the object image now tends to follow the true centroid image position. Sample results are shown in Figure 3.15 for a single run with initialization as specified by $\dot{\mathbf{x}}_0$ in Table 3.3. These results still show noisy behavior in estimation errors over the first 2 seconds, but of smaller magnitude than shown in Figure 3.11 of Case 3.2-3. After 2 seconds, y -position, translational velocity, and angular rate errors converge rapidly to zero. However, the x -position (depth) errors remain noisy throughout the run, and the angular position estimate after 3 seconds shows small oscillations of approximately 0.5° amplitude about the true value.

Figure 3.16 shows results for a 66 run Monte Carlo simulation in which 6 runs converged to alternate motion parameters and were removed. Again, these 6 runs did not diverge in the usual sense, but instead showed convergence to motion parameters with the negative rotational rate. Recall, however, that initialization errors for angular position and velocity are larger than those of Case 3.2-3 for which 44 runs converged to alternate motion and structural parameters. With the additional feature point, filter bias has been reduced substantially over the results shown for Case 3.2-3, and the RMSE for all states throughout the run is very close to Cramer-

Rao bounds. This improved performance results primarily from the inclusion of the additional feature point and can not be attributed entirely to known structure; motion parameter estimation for two feature points under known structure for this trajectory produces results that are almost identical to those shown for Case 3.2-3.

Figure 3.14: Image plane positions (units) of feature points for Case 3.3-1.

Figure 3.15: Sample results for Case 3.3-1.

2

MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS
STANDARD REFERENCE MATERIAL 1010a
(ANSI and ISO TEST CHART No. 2)

Figure 3.16: Monte Carlo simulation results for Case 3.3-1. All plots show — RMSE, - - - CRLB, and ME.

Table 3.4: Simulation parameters for Case 3.3-2

$\mathbf{x}$	$\mathbf{x}(0)$	$\hat{\mathbf{x}}_0$	$\sigma(\hat{\mathbf{x}}_0)$	$T = 1/30 \text{ Sec.}$
x	50.0	55.0	5.0	units
$\dot{x}$	0.5	0.0	3.0	units/s
y	-5.0	-3.0	5.0	units
$\dot{y}$	1.0	0.0	3.0	units/s
ψ	1.0	0.0	1.0	radians
w	-1.0	0.0	1.0	radians/s

Case 3.3-2: Four feature points with occlusion, central trajectory, high rotation, and low translation. The geometry of Figure 3.13 for the block motion demonstrated in Figure 3.1 was used to generate image positions of the four feature points with occlusion due to object motion. Image generation with occlusion is described in Appendix A. Structure of the translating and rotating block was defined by

$$S = \left\{ \begin{bmatrix} 10.0 \\ 5.0 \end{bmatrix}, \begin{bmatrix} 10.0 \\ -5.0 \end{bmatrix}, \begin{bmatrix} -10.0 \\ -5.0 \end{bmatrix}, \begin{bmatrix} -10.0 \\ 5.0 \end{bmatrix} \right\}. \quad (3.31)$$

Simulations were conducted for a trajectory similar to that used in Case 3.3-1 but with initialization parameters as shown in Table 3.4. The only differences between this trajectory and that of the previous case is in the structural parameter set S and the initial orientation $\psi(0)$. These parameters are consistent with further investigations of occlusion effects in Chapter 4.

Figure 3.17 shows true and sample images for this trajectory. For most of the run, three feature points are visible. However, at about 1, 2.6, 4.2, 5.8, 7.2, and 8.8 seconds, only two feature points are available. Sample results are shown in Figure 3.18 for a single run with initialization $\hat{\mathbf{x}}_0$ specified in Table 3.4. These results still demonstrate noisy behavior in the estimation errors over the first 2 seconds, but of smaller amplitude in comparison to Figure 3.15 of Case 3.3-1. After 2 seconds, y -position, translational velocity, and angular rate errors have almost converged. However, the x -position (depth) estimates remain slightly biased and noisy throughout the run, and the angular position estimate again shows small oscillations

about the true value, but also of smaller amplitude than shown in Figure 3.15 of Case 3.3-1.

Figure 3.19 shows the results for a 65 run Monte Carlo simulation in which 5 runs converged to alternate motion parameters and were removed. The filter bias in all states is practically negligible in this case, and the RMSE throughout the run is very close to the CRLB. The peak in angular position RMSE and CRLB at approximately 1 to 1.25 seconds corresponds to the first interval for which only two feature points are available. These results suggest that no serious degradation in filter performance should occur in the case of feature occlusion, provided at least two feature points are visible at all times.

Figure 3.17: Image plane positions (units) of feature points with occlusion for Case 3.3-2.

Figure 3.18: Sample results for Case 3.3-2.

Figure 3.19: Monte Carlo simulation results for Case 3.3-2. All plots show — RMSE, - - - CRLB, and ME.

Case 3.3-3: Cramer-Rao bound comparison for two, three, and four feature points without occlusion, and four feature points with occlusion. The Cramer-Rao bounds for four feature points with occlusion shown in Figure 3.19 of Case 3.3-2 appear to be lower, particularly in x -position (depth) and velocity, than those shown in Figure 3.16 of Case 3.3-1 for three feature points without occlusion. Figure 3.20 shows Cramer-Rao bounds with $P_0^{-1} = 0$ and the trajectory defined by the initial state $x(0)$ in Table 3.4 for two, three, and four feature points without occlusion, and for four feature points with occlusion. The block structure is identical to that of Case 3.3-2. These plots show a significant decrease in optimal estimation error bounds when the number of feature points is increased from two to three. However, as the number of feature points increases beyond three, performance bounds exhibit decreasing returns from the added computational burden. In the case of occlusion, at least two, but at most three feature points are visible in any given image. As a result, one would expect that the performance bounds under occlusion would lie between those for two and three feature points without occlusion. In the plots for y position and velocity, and angular position and velocity this is indeed the case. However, in the x position and velocity plots, performance bounds with occlusion lie consistently below those for three points without occlusion and in fact tend to follow bounds for four feature points without occlusion. Comparison of Figure 3.20 with the image of Figure 3.17 suggests that the sharp sawtooth ripples seen in the y position and velocity, and angular position and velocity performance bounds with occlusion correspond to times during which only two feature points are visible.

Figure 3.20: Cramer-Rao bound variation with number of feature points and occlusion. All plots show — four feature points with occlusion, and - - - four, three, and - . - . - two feature points without occlusion.

3.3.6 Discussion

A primary assumption of this investigation is that structure of the object is known in advance. Even though this assumption is certainly restrictive, many robotics applications could benefit from motion analysis for the known structure case. For example, industrial robots often operate in very controlled environments and on known objects, or a vision system could be used to observe the motion of a known robot arm and known objects in order to provide control signals to coordinate interaction between the manipulator and objects of interest. Dickmanns and Graefe [38], for example, exploit known object structure in some of their investigations of real-time vision-based control. Further developments in Chapter 4 for the planar motion case remove requirements of known structure. The constant translational and rotational velocity, and noiseless system (plant) model assumptions have been employed here to place emphasis on recursive estimation performance under the nonlinear perspective projection image formation (measurement) model, and to provide a simplification for demonstration purposes. Extensions of the system model to incorporate random disturbances and/or higher time derivatives of motion is straightforward, but is somewhat application specific.

Simulation results for this technique have demonstrated that significant improvements in performance can be realized if three feature points are employed as opposed to two. Increasing the number of feature points beyond three results in only slight reductions in performance bounds. In applications, therefore, decreasing returns in performance as the number of feature points increases could be exploited to maintain both sufficient accuracy and minimal computational complexity. This is particularly true under feature occlusion. Moreover, it may be beneficial to implement an active, time-varying selection of feature points for the measurement sequence according to the accuracy with which these points can be measured. Dickmanns and Graefe [38], and Schell and Dickmanns [41], for example, have applied this approach to autonomous vehicle docking and aircraft landing problems, respectively.

3.4 Chapter Summary

Simulation analysis of this chapter have investigated a special case of the planar motion problem in which the optical axis and central scan line of the imaging system lie in the plane of motion. Broida and Chellappa's approach to this problem in [24] provides a technique to recover motion and some structural parameters through noisy measurements of horizontal positions of image feature points. Simulation results for their technique using two feature points demonstrated satisfactory performance provided object rotational rates are relatively high. As rotational rates decrease, bounds on motion and structure estimation accuracy are seriously degraded. Moreover, the structural model employed in this approach does not fully exploit the rigid object assumption. This problem will be considered further in Chapter 4.

Under an assumption of known structure, a slightly different approach has been proposed for the planar motion case. The motion model developed for this case will be employed again in Chapter 4, but with arbitrary viewing angles off of the plane of motion. Simulation analysis for known structure and the specialized geometry considered in this chapter investigated filter performance with respect to the number of feature points available. If three feature points are available, filter performance is greatly improved in that the estimation RMS error follows Cramer-Rao bounds. Increasing the computational burden by considering more than three feature points, however, yields decreasing returns in terms of performance bounds.

The motion model of Section 3.3 is independent of the number of visible feature points due to the known structure assumption. This will not be the case in Chapter 4 which again considers structure estimation. It has been demonstrated, however, that the recursive structure of the extended Kalman filter provides natural solutions in the presence of occlusion in that the dimension of the measurement sequence need not remain constant. In fact the investigations of Section 3.3 suggest that no serious degradation in filter performance should be expected provided a sufficient number (two for the problem considered here) of feature points are visible at all times.

Chapter 4

Planar Motion, Arbitrary Viewing Angles

4.1 Introduction

The development of Chapter 3 was based on knowledge that the object was constrained to lie in a known plane and that objects were viewed from the plane of motion. These assumptions resulted in linear system (plant) models to describe object motion, and measurement models of reduced dimension since only horizontal positions of image feature points were considered.

This chapter examines a much broader class of motion analysis problems. In [27], Broida and Chellappa develop motion and measurement models which describe a three-dimensional object undergoing constant translational and rotational velocity with two-dimensional measurements taken from a monocular image sequence. Object rotation may be about an arbitrary axis in space and, therefore, need not have any direct relationship to translational motion. In order to eliminate an arbitrary scale factor associated with estimation of translational motion and object structure, these states have been referenced to the image plane through normalization by the unknown, time-varying depth of the object-centred reference frame. Rotational motion, which is not subject to arbitrary scaling, is modeled with quaternions in absolute form. The resulting dynamic model is bilinear in state variables. Structure of the rigid object is not modeled by constant parameters due to normalization by

time-varying depth. Their approach does not require prior information about object structure, however, depth information from an alternate source is required in order to recover absolute position, translational motion, and structure. Although this method may be theoretically appealing, it is not a realistic approach in applications which demand absolute translational motion and object structure. Depth measurements made available through an alternate source would most likely be contaminated with noise and, therefore, a more practical approach would be to remove the normalization of translational and structural parameters from the dynamic model and include depth measurements along with their uncertainty in the measurement model. In this case, not only would the dynamic model be simplified, but structural parameters would be modeled as constants allowing full exploitation of the rigid object assumption and removal of structure estimation once threshold confidence levels are attained. It should be noted that more recent developments of this technique by Young and Chellappa [30] for stereo imagery follow this general approach.

Most robotics applications require absolute position, motion, and structure of the object of interest with respect to a specified reference system. Section 4.3 considers a class of vision problems which can be described through motion on a known planar surface, and arbitrary camera positions and viewing angles above the plane of motion. Linear motion and structural models, and a nonlinear measurement model are proposed which form the basis for an extended Kalman filtering approach to recover absolute motion and structure of the object without the requirement for alternate sources of depth information. Knowledge of the plane of motion results in observability of all motion and structural parameters provided one observed feature point lies at a known height above the plane of motion. In the analysis presented here, one feature point is assumed to lie in the plane of motion. This assumption provides essential depth information to the estimation process for a monocular imaging system. Similar assumptions are employed by Dickmanns *et al.* [42] for obstacle avoidance in autonomous vehicle control with monocular imagery.

In contrast to the simulation analysis of Chapter 3, there were no cases during simulations of the present chapter in which either Broida and Chellappa's method or

the proposed technique for planar motion failed to converge to within a reasonable interval of the "true" motion parameters. However, Broida [31] has observed convergence to alternate motion and structural parameters for their general approach with a transparent object (no occlusion) assumption. Figure 4.1 shows a sequence of eight images for a rotating rectangular block which is constrained to lie on a plane and is observed from a position above the plane. Again, there are two interpretations for the structure of the object in each image frame: the first is consistent with the fact that the object is a rectangular block (all corners 90°), while the second interpretation views the object as a skewed block with one end smaller than the other. The apparent motion of either interpretation can be followed from one frame to the next (down the left column and up the right) through all eight frames.

The axis of rotation of the rectangular block (first interpretation) is normal to the planar surface on which the block is constrained to lie, and angular motion is counterclockwise looking down this axis from above. The apparent axis of rotation of the skewed block (second interpretation), on the other hand, is not perpendicular to the plane of motion. This demonstration suggests that imagery generated from one trajectory in state space again has two distinct interpretations under the viewing angles considered here, and that there exists a true object and object motion which is consistent with the skewed-block interpretation. Broida and Chellappa's parameterization of motion can describe either interpretation of object motion, which may suggest that both solutions lie in the state space associated with their problem formulation. The proposed technique for motion on a known plane, however, restricts the rotational axis to be perpendicular to this plane and hence can not model the rotational motion associated with the skewed-block interpretation.

Figure 4.2 shows the same image sequence as Figure 4.1, but with occlusion. In this case, there appears to be only one interpretation for object motion and structure. However, the absence of the second interpretation may be largely due to the ability of the human visual system to "filter out" any interpretation which is not consistent with the occlusion process. Feature point occlusion is treated in this investigation by simply continuing with parameter estimation, but without data from the occluded points. Hence the data set with feature point occlusion is contained in that obtained

Figure 4.1: Two interpretations of rotational 3D motion without occlusion. In each image frame, there are two interpretations of the object structure and the direction of the axis of rotational motion.

Figure 4.2: Rotational 3D motion without occlusion. In each image frame, there appears to be only one interpretation of object structure and motion.

without occlusion. This suggests that the second interpretation which is visible in Figure 4.1 is still a valid result even under occlusion, provided it can be generated by a trajectory in the state space. Unfortunately, these observations are merely suggestive and further analysis is required in order to establish uniqueness results.

Section 4.2 reviews the development of Broida and Chellappa's technique and presents results and performance assessments for simulated imagery. Sample and Monte Carlo simulation results are shown for the trajectory employed in [28], primarily to verify correct behavior of the filtering algorithm. Most states show RMS errors significantly above Cramer Rao bounds, particularly in quaternion estimation. As in investigations of Chapter 3, this system also requires prior knowledge concerning the presence or absence of rotational motion since an alternate approach must be taken in the case of zero angular velocity. It should be expected, therefore, that convergence rates will be reduced for smaller angular velocity. Moreover, the frequency of oscillation in quaternion estimation errors appears to be directly related to the magnitude of angular velocity. Sample and Monte Carlo simulation results are shown for a trajectory in which the object is constrained to lie on a planar surface in order to provide a performance comparison between Broida and Chellappa's method and the proposed technique for planar motion.

Simulation studies of the proposed technique include performance assessments for motion estimation under known structure, and for both motion and structure estimation. Artificial assumptions of transparent objects are not employed in the filter evaluation; only imagery with occlusion is considered. The proposed filter models structural parameters as constants which may be decoupled from the estimation process once threshold confidence levels are attained. In this case, motion and structure estimation is eventually reduced to motion estimation under (approximately) known structure. In addition, many robotics applications can be envisioned for the known structure case. Knowledge of structure as well as the plane of motion provides extensive information to the estimation process which shows excellent performance for this case. A natural concern in applications is the number of feature points that are required in order to give satisfactory performance. A comparison of Cramer Rao bounds demonstrates that only slight improvements in performance

can be expected when more than four feature points are considered. Just as in Broida and Chellappa's technique, the proposed method for planar motion requires prior knowledge concerning the presence or absence of rotational motion. Moreover, feature points are occluded over longer periods of time as rotational rates decrease. As a result, performance degradation should be expected for trajectories with small angular velocity. Sample and Monte Carlo simulation results for two planar motion trajectories, the first with low translational motion and fairly high angular rates, and the second with higher translation velocities and reduced angular velocity, demonstrate these properties of the proposed algorithm.

The following sections share a common structure similar to that employed in Chapter 3: the system geometry appropriate to the specific problem is described, followed by a development of system and measurement models, linearization of the models, and finally simulation results are presented. Again, simulation analysis of this chapter consider distinct "Cases" associated with a particular set of trajectory, structure, and stochastic parameters. Graphical results shown for each Case may include a representative image resulting from a particular trajectory, parameter estimation errors of a sample run, the approximate Cramer-Rao lower bounds (CRLB), and the root-mean-square-errors (RMSE) and mean errors (ME) of a 60 run Monte Carlo simulation. The ME and RMSE over 60 runs were computed as in Equations (3.1). Cramer-Rao bounds are computed from the true trajectory as shown in Chapter 2, Section 2.4.

4.2 Broida and Chellappa's Approach

Broida and Chellappa have extended their approach of [24] to more general viewing angles. This extended technique is introduced in [25], while more recent and formal developments are available in [28] which proposes a motion model for constant translational and rotational velocity. This section reviews the methods presented in [28], however, the notation has again been altered to reflect the definition of the camera-reference frame shown in Figure 2.1.

4.2.1 System Geometry

Figure 4.3 demonstrates a typical physical arrangement for the single camera system and moving object. Translational and rotational kinematics are described with respect to a stationary camera reference frame, F_C . In [28], Broida and Chellappa begin with general discussions for arbitrary motion, but restrict their development to unforced motion with constant translational and angular velocities. In this case, object rotational kinematics include the angular orientation of the object-centred reference frame F_O relative to F_C and the corresponding angular velocity about an axis through the origin of F_O . This leaves two degrees of freedom in the specification of F_O which must be rigidly attached to the object. One degree of freedom is along the axis of rotation, which is eliminated by assuming that one coordinate, here the z -coordinate, of one feature point is known relative to F_O . The second degree of freedom is an angular displacement about the rotational axis, which is treated by assuming perfect knowledge of the initial orientation of F_O relative to F_C (perfect alignment, for example). All feature points are expressed as constant vectors with respect to the object-centred frame. Translational kinematics are defined to be the position and constant velocity of the origin of F_O , which is not observed directly, expressed with respect to F_C . Again, a transparent object assumption is employed so that feature occlusion need not be considered.

4.2.2 Motion and Structure Model

Translational motion is modeled with standard rectilinear states, while rotational motion is described with quaternions. Under the assumption of constant angular velocity, the time propagation of the quaternion of rotation can be written in closed form as shown in Section 2.2.3, Equation (2.42).

With a monocular imaging system, object structure and 3D translational kinematics can only be recovered to within a scale factor unless some prior information about object structure or translational motion is assumed. Object rotational kinematics are not subject to this scale factor. In [28], states corresponding to object structure and translational motion have been referenced to the image plane through

Figure 4.3: System geometry for Broida and Chellappa's approach.

normalization by the unknown, time-varying depth of the origin of F_O .

The origin of F_O is described in F_C by $[x(t), y(t), z(t)]^T$ and has constant translational velocity $[\dot{x}, \dot{y}, \dot{z}]^T$. The orientation of F_O relative to F_C is described by the quaternion $\mathbf{q}(t) = [q_1(t), q_2(t), q_3(t), q_4(t)]^T$, where the ordering of $\mathbf{q}$ is as defined in (2.35). The constant angular velocity of F_O relative to F_C expressed in F_C is represented by $\mathbf{w} = [w_1, w_2, w_3]^T$. Object structure is modeled by a set of constant vectors in F_O as $\mathcal{S} = \{\mathbf{r}_O^i | i = 1, 2, \dots, N_p\}$, where each structure vector is expressed in component form as $\mathbf{r}_O^i = [r_{O,1}^i, r_{O,2}^i, r_{O,3}^i]^T$ and N_p is the number of feature points available. With this notation and the geometry of Figure 4.3, the state vector is defined as

$$\mathbf{x}(t) = \begin{bmatrix} y(t)/x(t) \\ z(t)/x(t) \\ \dot{x}/x(t) \\ \dot{y}/x(t) \\ \dot{z}/x(t) \\ q_1(t) \\ q_2(t) \\ q_3(t) \\ q_4(t) \\ w_1 \\ w_2 \\ w_3 \\ r_{O,1}^1/x(t) \\ r_{O,2}^1/x(t) \\ r_{O,3}^1/x(t) \\ \vdots \\ r_{O,1}^{N_p}/x(t) \\ r_{O,2}^{N_p}/x(t) \\ r_{O,3}^{N_p}/x(t) \end{bmatrix} \triangleq \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \\ s_8 \\ s_9 \\ s_{10} \\ s_{11} \\ s_{12} \\ s_{13} \\ s_{14} \\ s_{15} \\ \vdots \\ s_{3N_p+10} \\ s_{3N_p+11} \\ s_{3N_p+12} \end{bmatrix}. \quad (4.1)$$

Division of all translational and structural parameters by $x(t)$, which represents the depth of the object from the camera, results in a repeated normalization and elim-

inates an arbitrary scale factor associated with the estimation of these parameters. Moreover, states s_1 and s_2 are the (unobserved) image plane coordinates of the origin of F_O with unit focal length which may suggest a partial initialization procedure for the recursive estimation process.

The state vector in (4.1) can be differentiated with respect to time, with the aid of (2.38), to give a noise-free motion and structure model as

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} \dot{y}/x - (y/x)(\dot{x}/x) \\ \dot{z}/x - (z/x)(\dot{x}/x) \\ -(\dot{x}/x)^2 \\ -(\dot{y}/x)(\dot{x}/x) \\ -(\dot{z}/x)(\dot{x}/x) \\ 0.5(-w_3q_2 + w_2q_3 - w_1q_4) \\ 0.5(w_3q_1 - w_1q_3 - w_2q_4) \\ 0.5(-w_2q_1 + w_1q_2 - w_3q_4) \\ 0.5(w_1q_1 + w_2q_2 + w_3q_3) \\ 0 \\ 0 \\ 0 \\ -(r_{O,1}^1/x)(\dot{x}/x) \\ -(r_{O,2}^1/x)(\dot{x}/x) \\ -(r_{O,3}^1/x)(\dot{x}/x) \\ \vdots \\ -(r_{O,1}^{N_p}/x)(\dot{x}/x) \\ -(r_{O,2}^{N_p}/x)(\dot{x}/x) \\ -(r_{O,3}^{N_p}/x)(\dot{x}/x) \end{bmatrix} = \begin{bmatrix} s_4 - s_1s_3 \\ s_5 - s_2s_3 \\ -s_3^2 \\ -s_4s_3 \\ -s_5s_3 \\ 0.5(-s_{12}s_7 + s_{11}s_8 - s_{10}s_9) \\ 0.5(s_{12}s_8 - s_{10}s_8 - s_{11}s_9) \\ 0.5(-s_{11}s_6 + s_{10}s_7 - s_{12}s_9) \\ 0.5(s_{10}s_6 + s_{11}s_7 + s_{12}s_8) \\ 0 \\ 0 \\ 0 \\ -s_{13}s_3 \\ -s_{14}s_3 \\ -s_{15}s_3 \\ \vdots \\ -s_{3N_p+10}s_3 \\ -s_{3N_p+11}s_3 \\ -s_{3N_p+12}s_3 \end{bmatrix}, \quad (4.2)$$

where time dependency has been suppressed to simplify the notation, but can be inferred from (4.1). This continuous-time system model is bilinear in the elements of the state vector. Moreover, since the first two states, s_1 and s_2 , represent the image plane position of the origin of F_O , the first two elements of $\dot{\mathbf{x}}(t)$, namely $s_4 - s_1s_3$ and $s_5 - s_2s_3$, correspond to the optical flow in the image plane of the origin of F_O [31, page 74].

As in Broida and Chellappa's 2D technique reviewed in Chapter 3, the axis of rotation passes through the origin of F_O which undergoes constant translational velocity. This parameterization of object motion requires prior knowledge concerning the presence or absence of rotational motion. Degraded performance, therefore, should be expected as rotational rates decrease. In order to fix the origin of F_O on the rotational axis (line), the z_O -component of one feature point is assumed, without loss of generality, to be a known constant equal to zero. This means, for example, that s_{3N_p+12} can be removed from the state vector. However, the development in [28] continues with the full state vector corresponding to that defined in (4.1). In order to remain consistent with their development, except for the definition of F_C , the full state vector will be used in this review. Additional comments on this practice are included in Section 4.2.4.

Equation (4.2) with additive system noise can be written in the form of (2.70) as

$$\dot{\mathbf{x}}(t) = \mathbf{f}[\mathbf{x}(t), t] + \mathbf{G}(t)\mathbf{w}(t). \quad (4.3)$$

The Jacobian of $\mathbf{f}[\mathbf{x}(t), t]$, which is required in the continuous-discrete form of the extended Kalman filter reviewed in Section 2.3.3, is given by

$$\begin{aligned} \mathbf{F}[t; \mathbf{x}] &\triangleq \frac{\partial \mathbf{f}[\mathbf{x}, t]}{\partial \mathbf{x}} \\ &= \begin{bmatrix} \mathbf{F}_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{F}_2 & \mathbf{0} \\ \mathbf{F}_3 & \mathbf{0} & \mathbf{F}_4 \end{bmatrix}, \end{aligned} \quad (4.4)$$

where the matrices $\mathbf{F}_1 \in R^{5 \times 5}$, $\mathbf{F}_2 \in R^{7 \times 7}$, $\mathbf{F}_3 \in R^{3N_p \times 5}$, and $\mathbf{F}_4 \in R^{3N_p \times 3N_p}$ are defined, with the notation of (4.1) and (4.2), as follows:

$$\begin{aligned} \mathbf{F}_1 &\triangleq \left[\frac{\partial f_j[\mathbf{x}, t]}{\partial x_k} \right]_{j, k = 1, 2, \dots, 5}, \\ &= \begin{bmatrix} -s_3 & 0 & -s_1 & 1 & 0 \\ 0 & -s_3 & -s_2 & 0 & 1 \\ 0 & 0 & -2s_3 & 0 & 0 \\ 0 & 0 & -s_4 & -s_3 & 0 \\ 0 & 0 & -s_5 & 0 & -s_3 \end{bmatrix}; \end{aligned} \quad (4.5)$$

$$\begin{aligned}
\mathbf{F}_2 &\triangleq \left[\frac{\partial f_j[\mathbf{x}, t]}{\partial \mathbf{x}_k} \right] \quad j, k = 6, 7, \dots, 12, \\
&= 0.5 \begin{bmatrix} 0 & -s_{12} & s_{11} & -s_{10} & -s_9 & s_8 & -s_7 \\ s_{12} & 0 & -s_{10} & -s_{11} & -s_8 & -s_9 & s_6 \\ -s_{11} & s_{10} & 0 & -s_{12} & s_7 & -s_6 & -s_9 \\ s_{10} & s_{11} & s_{12} & 0 & s_6 & s_7 & s_8 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}; \quad (4.6)
\end{aligned}$$

$$\begin{aligned}
\mathbf{F}_3 &\triangleq \left[\frac{\partial f_j[\mathbf{x}, t]}{\partial \mathbf{x}_k} \right] \quad j = 13, 14, \dots, 3N_p + 12; \quad k = 1, 2, \dots, 5, \\
&= \begin{bmatrix} 0 & 0 & -s_{13} & 0 & 0 \\ 0 & 0 & -s_{14} & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & -s_{3N_p+12} & 0 & 0 \end{bmatrix}; \text{ and} \quad (4.7)
\end{aligned}$$

$$\begin{aligned}
\mathbf{F}_4 &\triangleq \left[\frac{\partial f_j[\mathbf{x}, t]}{\partial \mathbf{x}_k} \right] \quad j, k = 13, 14, \dots, 3N_p + 12, \\
&= -s_3 \mathbf{I}_{3N_p}, \quad (4.8)
\end{aligned}$$

where $\mathbf{I}_{3N_p}$ is the $3N_p \times 3N_p$ identity matrix. It is understood in the above that each $\mathbf{F}_i$ is evaluated at the most recent state estimate $\hat{\mathbf{x}}(k|k)$, $t_k \leq t \leq t_{k+1}$.

In the simulation analysis considered here, time propagation of the state estimate, from Equation (2.83), between measurement updates is approximated through first-order Euler quadrature of the noise free differential equation (4.3),

$$\hat{\mathbf{x}}(k+1|k) \approx \hat{\mathbf{x}}(k|k) + T_k \mathbf{f}[\hat{\mathbf{x}}(k|k)], \quad (4.9)$$

where $T_k = t_{k+1} - t_k$ is the k th sample interval. This approximation was found to yield better results than those obtained by employing an approximate state transition matrix of the form

$$\hat{\Phi}(k) \approx e^{\mathbf{F}[\hat{\mathbf{x}}(k|k)]T_k}. \quad (4.10)$$

However, simulation results consistent with those shown in [28] are obtained for the noise-free system case ($\mathbf{G}(t) \equiv \mathbf{0}$) with approximate time propagation of the

covariance matrix taken according to

$$\dot{\mathbf{P}}(k+1|k) \approx \hat{\Phi}(k)\dot{\mathbf{P}}(k|k)\hat{\Phi}^T(k), \quad (4.11)$$

where $\hat{\Phi}(k)$ is as shown in (4.10). This approximate time propagation of the state estimate and its covariance is consistent with the approach taken in [31]. It is anticipated, however, that further performance improvements may be obtained through more accurate numerical integration of the system differential equation and the Riccati covariance equation at the expense of increased computational complexity.

4.2.3 Measurement Model

The measurement model is assumed to be given by perspective projection which, with the present notation and the development of Section 2.2.1, can be written as

$$\mathbf{z}(k) = \mathbf{h}[\mathbf{x}(k)] + \mathbf{v}(k), \quad (4.12)$$

where

$$\mathbf{h}[\mathbf{x}(k)] = [\mathbf{h}^i[\mathbf{x}(k)]], \quad i = 1, 2, \dots, N_p, \quad (4.13)$$

and

$$\mathbf{h}^i[\mathbf{x}] = \begin{bmatrix} \frac{s_1 + [0, 1, 0]\mathbf{I}_C^O(\mathbf{q})\mathbf{r}_O^i/x}{1 + [1, 0, 0]\mathbf{I}_C^O(\mathbf{q})\mathbf{r}_O^i/x} \\ \frac{s_2 + [0, 0, 1]\mathbf{I}_C^O(\mathbf{q})\mathbf{r}_O^i/x}{1 + [1, 0, 0]\mathbf{I}_C^O(\mathbf{q})\mathbf{r}_O^i/x} \end{bmatrix}. \quad (4.14)$$

The normalized structure vectors $\mathbf{r}_O^i/x$ are elements of the state vector $\mathbf{x}$:

$$\mathbf{r}_O^i/x = \begin{bmatrix} s_{3i+10} \\ s_{3i+11} \\ s_{3i+12} \end{bmatrix}. \quad (4.15)$$

The rotation matrix $\mathbf{I}_C^O(\mathbf{q})$ is defined in (2.37) as

$$\mathbf{I}_C^O(\mathbf{q}) = \begin{bmatrix} q_1^2 - q_2^2 - q_3^2 + q_4^2 & 2(q_1q_2 + q_3q_4) & 2(q_1q_3 - q_2q_4) \\ 2(q_1q_2 - q_3q_4) & -q_1^2 + q_2^2 - q_3^2 + q_4^2 & 2(q_2q_3 + q_1q_4) \\ 2(q_1q_3 + q_2q_4) & 2(q_2q_3 - q_1q_4) & -q_1^2 - q_2^2 + q_3^2 + q_4^2 \end{bmatrix}, \quad (4.16)$$

where $[q_1, q_2, q_3, q_4] \equiv [s_6, s_7, s_8, s_9]$. The measurement model is then given by (4.12), with (4.13)-(4.16).

Implementation of the continuous-discrete EKF requires the computation of $\mathbf{I}_C^O(\mathbf{q})$ in (4.16) and $\mathbf{h}[\mathbf{x}]$ in (4.13) as well as the matrix of partial derivatives,

$$\mathbf{H} \triangleq \frac{\partial \mathbf{h}[\mathbf{x}]}{\partial \mathbf{x}}, \quad (4.17)$$

from the most recent state estimate, $\hat{\mathbf{x}}(k|k-1)$. In the following development, the “ $\cdot$ ” notation is omitted, but it is understood that all quantities are computed from this state estimate. Since $\mathbf{h}[\mathbf{x}]$ must be computed, it is reasonable to assume that

$$[\mathbf{r}_O^i]_{C/x} = \mathbf{I}_C^O(\mathbf{q})\mathbf{r}_O^i/x, \text{ and} \quad (4.18)$$

$$\mathbf{r}_C^i/x = \begin{bmatrix} 1 \\ y/x \\ z/x \end{bmatrix} + [\mathbf{r}_O^i]_{C/x} \quad (4.19)$$

are available. The computation of $\mathbf{H}$ is further simplified with the formation of four matrices

$$\mathbf{A}_k \triangleq \frac{\partial \mathbf{I}_C^O(\mathbf{q})}{\partial q_k}, \quad k = 1, \dots, 4, \quad (4.20)$$

which are given by

$$\mathbf{A}_1 = 2 \begin{bmatrix} q_1 & q_2 & q_3 \\ q_2 & -q_1 & q_4 \\ q_3 & -q_4 & -q_1 \end{bmatrix}, \quad (4.21)$$

$$\mathbf{A}_2 = 2 \begin{bmatrix} -q_2 & q_1 & q_4 \\ q_1 & q_2 & q_3 \\ q_4 & q_3 & -q_2 \end{bmatrix}, \quad (4.22)$$

$$\mathbf{A}_3 = 2 \begin{bmatrix} -q_3 & q_4 & q_1 \\ -q_4 & -q_3 & q_2 \\ q_1 & q_2 & -q_3 \end{bmatrix}, \text{ and} \quad (4.23)$$

$$\mathbf{A}_4 = 2 \begin{bmatrix} q_4 & q_3 & -q_2 \\ -q_3 & q_4 & q_1 \\ q_2 & -q_1 & q_4 \end{bmatrix}. \quad (4.24)$$

In this case, $\mathbf{H} \in R^{2N_p \times 3N_p+12}$ can be written as

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}^1 \\ \mathbf{H}^2 \\ \vdots \\ \mathbf{H}^{N_p} \end{bmatrix} = \begin{bmatrix} \mathbf{P}^1 & \mathbf{0} & \mathbf{W}^1 & \mathbf{0} & \mathbf{S}^1 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{P}^2 & \mathbf{0} & \mathbf{W}^2 & \mathbf{0} & \mathbf{0} & \mathbf{S}^2 & \dots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{P}^{N_p} & \mathbf{0} & \mathbf{W}^{N_p} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{S}^{N_p} \end{bmatrix}, \quad (4.25)$$

where each $\mathbf{H}^i \in R^{2 \times 3N_p+12}$, $i = 1, 2, \dots, N_p$, is defined as

$$\mathbf{H}^i \triangleq \frac{\partial \mathbf{h}^i[\mathbf{x}]}{\partial \mathbf{x}}. \quad (4.26)$$

The submatrices $\mathbf{P}^i \in R^{2 \times 2}$, $\mathbf{W}^i \in R^{2 \times 4}$, and $\mathbf{S}^i \in R^{2 \times 3}$ are computed, with further reduction of the corresponding representations in [28], as follows:

$$\begin{aligned} \mathbf{P}^i &\triangleq \left[\frac{\partial \mathbf{h}_j^i[\mathbf{x}]}{\partial \mathbf{x}_k} \right], \quad j, k = 1, 2 \\ &= \mathbf{I}_2 \left(\frac{1}{r_{C,1}^i/x} \right); \end{aligned} \quad (4.27)$$

$$\begin{aligned} \mathbf{W}^i &\triangleq \left[\frac{\partial \mathbf{h}_j^i[\mathbf{x}]}{\partial q_k} \right], \quad j = 1, 2; \quad k = 1, 2, 3, 4 \\ &= \left[\frac{(\mathbf{r}_O^i/x)^T \mathbf{A}_k^T \mathbf{B}_j (\mathbf{r}_C^i/x)}{(r_{C,1}^i/x)^2} \right], \end{aligned} \quad (4.28)$$

where $\mathbf{B}_1 \triangleq \mathbf{k}_C^i$ and $\mathbf{B}_2 \triangleq -\mathbf{j}_C^i$;

$$\begin{aligned} \mathbf{S}^i &\triangleq \left[\frac{\partial \mathbf{h}_j^i[\mathbf{x}]}{\partial (r_{O,k}^i/x)} \right], \quad j = 1, 2; \quad k = 1, 2, 3 \\ &= \left[\frac{(r_{C,1}^i/x) l_{j+1,k} - (r_{C,j+1}^i/x) l_{1,k}}{(r_{C,1}^i/x)^2} \right], \end{aligned} \quad (4.29)$$

where $\mathbf{I}_C^O \triangleq [l_{mn}]$ $m, n = 1, 2, 3$.

In (4.28), $\mathbf{k}_C \triangleq [0, 0, 1]^T$ and $\mathbf{j}_C \triangleq [0, 1, 0]^T$, and the notation $\mathbf{k}_C^i$ and $\mathbf{j}_C^i$ denote

the matrix cross product operators, defined in (2.18), associated with $\mathbf{k}_C$ and $\mathbf{j}_C$, respectively.

Following each measurement update, a quaternion normalization step is employed through division by its Euclidean norm. In [47], it is shown that this normalization step does not degrade the estimation of the quaternion, but instead forces the "estimation" of a unit quaternion at each measurement update which ensures that the resulting rotation matrix is orthogonal.

With this measurement model, each sampled image provides $2N_p$ equations in $3N_p + 11$ unknowns, since one coordinate of one feature point is assumed to be known. For example, with four feature points of interest ($N_p = 4$), 23 state variables are employed with each image providing 8 equations. In this case, at least 3 image samples are required to obtain more equations than unknowns.

4.2.4 Results and Performance With Simulated Imagery

The geometry of Figure 4.3 was used to generate sequences of image feature points with correspondence information for a translating and rotating block. Details of the software employed in this study are given in Appendix A. Figure 4.4 shows the orientation of the object-centred frame on the block and demonstrates the labeling of feature points for discussions which follow. If object motion is truly unforced, the constant rotational axis will pass through the object's centre-of-gravity (CG) which translates with constant velocity. The origin of the object-centred frame, shown in Figure 4.4 to lie on the apparent base of the block, is a point which moves with constant translational velocity for all simulations of this chapter. Hence rotational motion is about an axis which passes through this point.

In [28], Broida and Chellappa employ a transparent object assumption and use four corners of a block as feature points, but do not indicate which corners are used. This investigation employs four feature points, p^1 , p^3 , p^5 , and p^7 which are assumed to be available in each image without occlusion. The occlusion problem is studied in Section 4.3 for motion on a known planar surface. Two different block sizes are used in the following simulation studies: the first is consistent with simulations in

Figure 4.4: Object with feature labels used in simulation studies.

[28], while the second is identical to that employed for analysis of the planar motion case in Section 4.3. However, the size of the block in the image is approximately the same in either case since different depths from the camera are employed.

In all cases considered here, a pseudo random Gaussian noise sequence with standard deviation $\sigma_v = 0.02$ units was added to true feature point positions in the image. This noise level represents quantities from 5% to 10% of the maximum size of the object in the image over the trajectory (when the object is closest to the camera). In all trajectories, the object image size decreases as the object moves away from the camera giving rise to increased measurement noise levels in comparison to the object image size. Filter performance for different noise levels has been adequately studied in [31] and, as discussed in Section 3.2, is fairly accurately predicted by Cramer-Rao bound analysis. In this simulation study, measurement noise levels are fixed at the moderately high value of $\sigma_v = 0.02$ units and results are produced for the case in which the object is constrained to lie on a planar surface to facilitate comparisons between Broida and Chellappa's approach to the proposed technique for planar motion in Section 4.3.

As noted in [28], the standard extended Kalman filter does not yield satisfactory results in this estimation problem. The continuous-discrete form of the extended Kalman filter, reviewed in Section 2.3.3, with local iterations, discussed in Section 2.3.4, is employed with a maximum of 10 iterations per measurement update. Iteration is terminated if the maximum element of the difference in successive iterates is less than 0.001. During the first 10 to 20 image samples, 10 iterations were often required. However, during the later portion of each trajectory only 3 or 4 iterations were sufficient.

Simulation studies for this estimation technique include the following two cases:

Case 4.2-1 Broida and Chellappa's simulation in [28] is reproduced, to the extent possible, primarily to verify correct behavior of the filtering algorithm and to set the context for further investigation. Monte Carlo simulation results are also shown and compared to Cramer-Rao bounds.

Case 4.2-2 The object is constrained to lie on a planar surface with true motion

parameters defined with respect to an earth-fixed frame in the plane of motion. The initial position and orientation of the object, and its constant translational and rotational velocity are transformed to equivalent representations in the camera frame in order to initialize the estimation algorithm. This trajectory has low translational rates with fairly high rotational rates and is employed again in Section 4.3.

This investigation concludes with a summary of results and a general discussion of Broida and Chellappa's technique.

Case 4.2-1 Broida and Chellappa's trajectory. The true object trajectory is equivalent to that for which results are shown in [28], however, motion parameters have again been altered to reflect the orientation of the camera reference frame defined in Figure 2.1. The block is a cube of sides four units so that the four feature points p^1 , p^3 , p^5 , and p^7 define the structure set

$$S = \left\{ \begin{bmatrix} 2 \\ 2 \\ 0 \end{bmatrix}, \begin{bmatrix} -2 \\ 2 \\ -4 \end{bmatrix}, \begin{bmatrix} -2 \\ -2 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 \\ -2 \\ -4 \end{bmatrix} \right\}. \quad (4.30)$$

Table 4.1 lists the simulation parameters for this case. The true initial depth of the object is 10 units with a depth velocity (in the x_C direction) of 0.15 units/s. An object of width 4 units placed 10 units from the camera produces, approximately, an image of the object with width 0.4 units. In this case the added noise levels of $\sigma_v = 0.02$ units represent quantities on the order of 5% of the object image size. A 100 second time interval is employed with a sample period of $T = 1$ second. Table 4.1 gives the true initial parameters $\mathbf{x}(0)$, a sample initialization $\hat{\mathbf{x}}(0)$, and the assumed standard deviation of initialization errors $\sigma(\hat{\mathbf{x}}_0)$ employed in a 60 run Monte Carlo simulation.

Figure 4.5 demonstrates the true and a sample noisy image sequence for this case over 100 seconds. The object translates 15 units in depth and approximately 32 units parallel to the image plane while completing slightly more than 5.5 revolutions. Figure 4.6 shows sample results for this trajectory when the filter was initialized with $\hat{\mathbf{x}}(0|-1) = \hat{\mathbf{x}}_0$ given in Table 4.1. Initialization values are not available in [28],

Table 4.1: Simulation parameters for Case 4.2-1

Parameter	$\mathbf{x}$	$\mathbf{x}(0)$	$\dot{\mathbf{x}}_0$	$\sigma(\dot{\mathbf{x}}_0)$	$T = 1 \text{ Sec.}$
True Depth (x)	—	10.0	—	—	Units
Depth Vel. $\dot{x}$	—	0.15	—	—	Units/s
y/x	s_1	0.0	0.1	0.1	
z/x	s_2	0.0	0.1	0.1	
$\dot{x}/x$	s_3	0.015	0.0	0.2	1/s
$\dot{y}/x$	s_4	0.025	0.0	0.2	1/s
$\dot{z}/x$	s_5	-0.02	0.0	0.2	1/s
q_1	s_6	0.0	0.0	0.0	
q_2	s_7	0.0	0.0	0.0	
q_3	s_8	0.0	0.0	0.0	
q_4	s_9	1.0	1.0	0.0	
w_1	s_{10}	0.2	0.1	0.2	radians/s
w_2	s_{11}	0.2	0.3	0.2	radians/s
w_3	s_{12}	0.2	0.15	0.2	radians/s
$\mathbf{r}_O^i/x$	$s_{13} \dots$	S/depth	0.0	$2/\text{depth}$	
$i = 1 \dots N_p$	s_{3N_p+11}	Eqn.4.30			

however, those employed here have reasonably small errors. The initial covariance matrix (also not available in [28]) was taken as $\hat{\mathbf{P}}(0| - 1) = \mathbf{P}_0$ where

$$\mathbf{P}_0 = \text{diag}[\sigma^2(\dot{\mathbf{x}}_0)], \quad (4.31)$$

with $\sigma(\dot{\mathbf{x}}_0)$ as shown in Table 4.1. It should be noted that the errors for translational motion and structure shown in Figure 4.6 are normalized by the time-varying depth of the origin of F_O which in this case is increasing with time. Normalized position, velocity, and structure errors, as well as angular velocity errors demonstrate satisfactory convergence after about 20 seconds (samples). The quaternion errors exhibit oscillations of small and decreasing amplitude about the true values.

Figures 4.7, 4.8, 4.9, and 4.10 show results of a 60 run Monte Carlo simulation with filter initialization taken in accordance with (2.92) as $\dot{\mathbf{x}}(0| - 1) \sim N(\mathbf{x}(0), \mathbf{P}_0)$,

where $\mathbf{P}_0$ is defined in (4.31). In this case, initial translational velocity estimation errors are somewhat high whereas position errors are relatively low. The initial quaternion is known precisely as indicated by the zero standard deviations on quaternion initialization errors. Angular velocity and structure errors have been set high enough so that zero values are within one standard deviation of the true parameters. Figure 4.7 shows normalized position and velocity root-mean-square-errors (RMSE), Cramer Rao bounds (CRLB), and mean errors (ME), as well as true depth of the trajectory. Unlike simulations of the previous chapter, there were no cases in which the filter failed to converge to within reasonable intervals of the true motion parameters. The RMSE of normalized position errors for this trajectory lie slightly above the CRLB and in fact tend to increase towards the end of the trajectory. Estimation of normalized velocity, on the other hand, is very good with apparent convergence within about 15 seconds (samples).

Figure 4.8 shows the RMSE, CRLB, and ME for quaternions and components of angular velocity as well as the root-mean-square and mean values of the convergence index $J(k)$ for the rotation matrix formed from the estimated quaternions and defined in (2.44). The ME for quaternion estimation shows small oscillations which results in poor performance as indicated by the RMSE in comparison to the approximate lower bounds. The true maximum value of q_1 , q_2 and q_3 is given from (2.36) and (2.43) as 0.577 which implies that errors in the first three quaternion elements at the end of the run are on the order of 10%. The true angular velocity given in Table 4.1 indicates that the object completes one full revolution about every 18.14 seconds. The half period of oscillations in the quaternion ME trace is also approximately 18 seconds. The maximum error in q_4 occurs when the angle relating F_O to F_C is an odd integer multiple of π , that is when the cosine of half of this angle equals zero. Maximum errors in q_1 , q_2 , and q_3 , on the other hand, tend to occur for even integer multiples of π , i.e., when the sine of half this angle equals zero. However, errors in the first three quaternion elements are also dependent on angular velocity estimation errors which tend to be biased, particularly in w_1 and w_3 . The convergence index $J(k)$ for the rotation operator formed from the quaternion estimates also shows oscillations of decreasing amplitude with approximately

10% RMS error at the end of the run (recall that $0 \leq J(k) \leq 1$).

Figures 4.9 and 4.10 demonstrate filter performance in the estimation of normalized structural parameters. At the end of the run, normalized x_O - and y_O -coordinates have a true value of 0.08 while z_O/depth for p^3 and p^7 is 0.16 and those for p^1 and p^5 are always zero. Most normalized structural parameter estimates have a final RMS error of approximately 0.01 which represents a 12% error for x - and y -coordinates. However, the x_O/depth coordinates of p^3 and p^7 have final RMS errors of almost 25% and 40%, respectively. In general the convergence of structural estimates is somewhat slow and exhibits significant bias over the first 25 to 50 seconds of the trajectory.

Figure 4.5: True and noisy image sequences for Case 4.2-1.

Figure 4.6: Sample results for Case 4.2-1.

Figure 4.7: Monte Carlo simulation results for position and velocity of Case 4.2-1. All plots except true depth show — RMSE, - - - CRLB, and ME.

Figure 4.8: Monte Carlo simulation results for orientation and angular velocity of Case 4.2-1. All plots except J show — RMSE, - - - CRLB, and ME. Convergence index plot shows — RMSE, and - - - ME.

Figure 4.9: Monte Carlo simulation results for structure of points p^3 and p^7 of Case 4.2-1. All plots show — RMSE, - - - CRLB, and ME.

Figure 4.10: Monte Carlo simulation results for structure of points p^1 and p^5 of Case 4.2-1. All plots show — RMSE, - - - CRLB, and ME.

Case 4.2-2 Motion on a planar surface. In this case the object is constrained to lie on a planar surface and motion is initially described with respect to an earth-fixed reference frame attached to the plane of motion (refer to Figure 4.17 of Section 4.3). The known position and orientation of the stationary camera-centred frame relative to this earth-fixed frame is used to transform initial conditions given relative to F_E into equivalent parameters relative to F_C . With the notation of Section 2.2.2, initial position, translational velocity, and rotational velocity are easily expressed with respect to F_C through the change-of-basis transformation $\mathbf{I}_C^O = \mathbf{I}_C^E \mathbf{I}_E^O$ and translations $\mathbf{T}_O^E$ and $\mathbf{T}_E^C$:

1. Initial position $[\mathbf{T}_O^E]_E$ given,

$$[\mathbf{T}_O^C]_C = [\mathbf{T}_E^C]_C + \mathbf{I}_C^E [\mathbf{T}_O^E]_E; \quad (4.32)$$

2. Initial velocity $[\mathbf{V}_E^O]_E$ (read as "the velocity of F_O relative to F_E written with respect to F_E ") given, and provided F_C is stationary relative to F_E ,

$$[\mathbf{V}_C^O]_C = \mathbf{I}_C^E [\mathbf{V}_E^O]_E; \quad (4.33)$$

3. Initial angular velocity $[\mathbf{w}_E^O]_E$ given, and again provided F_C is stationary relative to F_E ,

$$[\mathbf{w}_C^O]_C = \mathbf{I}_C^E [\mathbf{w}_E^O]_E; \quad (4.34)$$

4. Initial orientation of F_O relative to F_E is given by the transformation $\mathbf{I}_E^O$. First compute $\mathbf{I}_C^O = \mathbf{I}_C^E \mathbf{I}_E^O$. The initial quaternion is computed from (2.21), (2.22), (2.36), and (2.43) as

$$\begin{aligned} \gamma &= \arccos \left(\frac{1}{2} [\text{Tr} \{ \mathbf{I}_C^O \} - 1] \right) \\ \xi_C^1 &= \frac{1}{2 \sin(\gamma)} (\mathbf{I}_C^O - (\mathbf{I}_C^O)^T) \\ \mathbf{q}(0) &= \left[-\xi_{C,1} \sin\left(\frac{\gamma}{2}\right), -\xi_{C,2} \sin\left(\frac{\gamma}{2}\right), -\xi_{C,3} \sin\left(\frac{\gamma}{2}\right), \cos\left(\frac{\gamma}{2}\right) \right]^T. \end{aligned} \quad (4.35)$$

In the quaternion initialization of item 4 above, γ and ξ_C are the angle and axis of rotation, respectively, used in the transformation from F_O to F_C . In the present

analysis, the quaternion of rotation is defined as that representing the transformation from F_C to F_O hence the "complex conjugate" quaternion, formed by taking the negative of the first three elements of q , is required as shown in the formation of $q(0)$.

The object, in this case, is defined by a block which is twice as long (y_O direction) as it is high (z_O) and wide (x_O). Four feature points p^1 , p^3 , p^5 , and p^7 define the structure set

$$S = \left\{ \begin{bmatrix} 5 \\ 10 \\ 0 \end{bmatrix}, \begin{bmatrix} -5 \\ 10 \\ -10 \end{bmatrix}, \begin{bmatrix} -5 \\ -10 \\ 0 \end{bmatrix}, \begin{bmatrix} 5 \\ -10 \\ -10 \end{bmatrix} \right\}. \quad (4.36)$$

Table 4.2 lists simulation parameters for this case derived through the initialization procedure described above. The true initial depth of the object is approximately 50 units with a depth velocity (in the x_C direction) of 0.75 units/s. The initial object image width is now from 0.2 to 0.4 units. In this case the added noise levels of $\sigma_v = 0.02$ units represent quantities on the order of 5% to 10% of the object image size when the object is closest to the camera. A 10 second time interval is employed with a sample period of $T = 0.1$ seconds. Table 4.2 gives the true initial parameters $x(0)$, a sample initialization $\hat{x}(0)$, and the assumed standard deviation of initialization errors $\sigma(\hat{x}_0)$ employed in a 60 run Monte Carlo simulation.

Figure 4.11 shows the true and a sample noisy image sequence for this case over 10 seconds. The object translates 7.5 units in depth and approximately 8 units parallel to the image plane while completing 1.5 revolutions. Measurement noise levels in this case appear to be more significant than those shown in Figure 4.5 due to the smaller inter-sample motion of the object. Figure 4.12 shows sample results for this trajectory when the filter was initialized with $\hat{x}(0|-1) = \hat{x}_0$ given in Table 4.2. Again, the errors for translational motion and structure shown in Figure 4.12 are normalized by the time-varying depth of the origin of F_O which is increasing with time. After some initial erratic behavior, all states appear to have converged after about 5 seconds (50 samples). However, some bias is seen in the normalized y -position errors and p^5 and p^7 structure errors.

Figures 4.13, 4.14, 4.15, and 4.16 show results of a 60 run Monte Carlo simulation

Table 4.2: Simulation parameters for Case 4.2-2

Parameter	x	$x(0)$	$\hat{x}_0$	$\sigma(\hat{x}_0)$	$T = 1/10$ Sec.
True Depth (x)	—	48.71	—	—	Units
Depth Vel. $\dot{x}$	—	0.75	—	—	Units/s
y/x	s_1	-0.0726	-0.1342	0.1	
z/x	s_2	-0.1291	-0.0675	0.1	
$\dot{x}/x$	s_3	0.0154	0.0	0.05	1/s
$\dot{y}/x$	s_4	0.0073	0.0	0.05	1/s
$\dot{z}/x$	s_5	-0.0154	0.0	0.05	1/s
q_1	s_6	-0.0410	-0.0410	0.0	
q_2	s_7	-0.3805	-0.3805	0.0	
q_3	s_8	-0.0989	-0.0989	0.0	
q_4	s_9	0.9186	0.9186	0.0	
w_1	s_{10}	-0.7071	-0.8071	0.2	radians/s
w_2	s_{11}	0.0	0.1	0.2	radians/s
w_3	s_{12}	-0.7071	-0.5571	0.2	radians/s
r_O^i/x	$s_{13} \dots$	S/depth	0.0	$5/\text{depth}$	
$i = 1 \dots N_p$	s_{3N_p+11}	Eqn.4.36			

with filter initialization taken in accordance with (2.92) as $\hat{x}(0|-1) \sim N(x(0), P_0)$ where P_0 is defined in (4.31) and $\sigma(\hat{x}_0)$ is as shown in Table 4.2. In this case, initial translational velocity errors are lower than those of Case 4.2-1. Again, the initial quaternion is known precisely as indicated by the zero standard deviations on quaternion initialization errors. Angular velocity and structure errors have been set high enough so that zero values are within one standard deviation of the true parameters. Figure 4.13 shows normalized position and velocity root-mean-square-error (RMSE), Cramer Rao bounds (CRLB), and mean errors (ME), as well as true depth of the trajectory. There were no runs for which the filter failed to converge to within reasonable intervals of the true motion parameters. Estimation of normalized position for this trajectory is slightly biased over the first 4 seconds which yields

RMSE results above the CRLB. Estimation of normalized velocity, although slightly biased over the first 4 seconds, shows RMS errors very close to Cramer Rao bounds.

Figure 4.14 shows the RMSE, CRLB, and ME for the quaternions and components of angular velocity as well as the root-mean-square and mean values of the convergence index $J(k)$ defined in (2.44). The ME for quaternion estimation again shows oscillations which results in poor performance as indicated by the RMSE in comparison to the approximate lower bounds. The true angular velocity given in Table 4.2 indicates that the object completes one full revolution about every 6.28 seconds. As noted in Case 4.2-1, the half period of oscillations in quaternion errors is also approximately 6 seconds. Note also that the CRLB for q_2 and both the RMSE and CRLB of q_4 are almost exactly zero at about 6 seconds, or when the object has made one complete revolution. Angular velocity components w_1 and w_3 tend to show biased estimation errors with poor RMSE performance throughout the run. This behavior was also noted in Figure 4.8 of Case 4.2-1, so this behavior does not appear to be strictly dependent on the trajectory. The convergence index $J(k)$ for the rotation operator formed from the quaternion estimates also shows oscillations of decreasing amplitude with approximately 5% RMS error at the end of the run.

Figures 4.15 and 4.16 demonstrate filter performance in the estimation of normalized structural parameters. At the end of the run, normalized x_O -coordinates have a true value of 0.18, y_O/depth is 0.35, and z_O/depth for p^3 and p^7 is also 0.35. Most normalized structural parameter estimates have a final RMS error of approximately 0.01 which represents errors on the order of 3 to 5%. RMSE performance in this case show a slight improvement over those for Case 4.2-1 (which had the same number of observation events) near the end of the run. In general the convergence of structural estimates is slow in comparison to performance bounds and exhibits significant bias over the first 5 seconds.

Figure 4.11: True and noisy image sequences for Case 4.2-2.

Figure 4.12: Sample results for Case 4.2-2.

Figure 4.13: Monte Carlo simulation results for position and velocity of Case 4.2-2. All plots except depth show — RMSE, - - - CRLB, and ME.

Figure 4.14: Monte Carlo simulation results for orientation and angular velocity of Case 4.2-2. All plots except J show — RMSE, --- CRLB, and ME. Convergence index plot shows — RMSE, and --- ME.

Figure 4.15: Monte Carlo simulation results for structure of points p^3 and p^7 of Case 4.2-2. All plots show — RMSE, --- CRLB, and ME.

Figure 4.16: Monte Carlo simulation results for structure of points p^1 and p^5 of Case 4.2-2. All plots show — RMSE, --- CRLB, and ME.

4.2.5 Discussion

Broida and Chellappa's approach to motion and structure estimation through measurements taken in noisy monocular image sequences can only yield absolute translational and structural parameters if accurate depth information is available. This is an immediate consequence of normalization of all translational and structural parameters by the time varying depth of the origin of the object-centred reference frame. A natural concern in applications which demand absolute position, velocity, and structure estimation is just how such depth information could be obtained. The origin of F_O need not lie at the centre-of-gravity of the object. In fact, the act of setting one F_O -coordinate of one feature point equal to zero, together with the assumption that F_O and F_C are initially known to be perfectly aligned, places the origin of F_O at a point on the (constant) rotational axis such that this special feature point lies in a coordinate plane. In this case, it is almost certain that the origin of F_O does not coincide with the CG position and, moreover, need not lie on or in the object at all. If depth information is provided to this algorithm through an independent sensor, for example radar, sonar, or laser ranging systems, then naturally such measurements will be contaminated with noise, but in addition will very likely give biased estimates of the depth of the origin of F_O . The level of bias in these depth estimates could be on the order of the dimensions of the object of interest or higher. If noisy depth measurements are available, a more realistic approach would be to remove the normalization of translational and structural parameters from the dynamic model and include these depth measurements along with their uncertainty in the measurement model. In this case, not only would the dynamic model be simplified, but structural parameters would be modeled as constants allowing full exploitation of the rigid object assumption and removal of structural estimation once threshold confidence levels are attained. More recent developments of this technique by Young and Chellappa [30] for stereo imagery follow this general approach.

An assumption of absolute prior knowledge of the initial quaternion of rotation and knowledge of one F_O -coordinate of one feature point have been employed to fix the object-centred reference frame on the object. This raises two concerns in real

applications: 1) does precise *a priori* knowledge of the initial quaternion provide sufficient information to eliminate an angular degree of freedom about the rotational axis, particularly in the presence of initial angular velocity uncertainty and system (plant) noise; and 2) how is estimation performance degraded in the presence of feature occlusion, particularly when the special feature point whose one coordinate is known becomes occluded from view. These concerns have not been adequately addressed in [31, 28] nor in this investigation. It is anticipated, however, that unmodeled dynamics, system noise, and feature point occlusion will result in degraded filter performance, especially in quaternion estimation.

In general satisfactory performance was shown for the two cases considered. Although some bias is seen during initial iterations of the filter, normalized position, velocity, and structure demonstrate acceptable convergence. Quaternion estimation errors, however, show oscillatory behavior of decreasing amplitude and may require a large number of image samples to attain a high confidence level. Angular velocity estimation showed a constant bias, especially in the x_E - and z_E -components. This may be due partially to the noise-free system model; some improvements might be achieved through "filter tuning" by incorporating system (plant) noise levels in order to maintain higher filter gains over a large number of image frames.

4.3 Motion On A Known Planar Surface

The previous section has reviewed Broida and Chellappa's method of motion and structure estimation from monocular image sequences. Although satisfactory filter performance has been demonstrated for two object trajectories, absolute position, translational motion, and structural parameters cannot be recovered without accurate depth information. In many robotics applications, however, machine vision problems can be described through motion on a known plane with arbitrary camera positions and viewing angles above the plane of motion. In this case, a ray from the centre of projection through a given point in the image intersects the plane in a single point. Neglecting all uncertainties, this means that the distance from the centre of projection to the point of intersection can be computed from the image

plane coordinates of the point and an analytical expression for the plane relative to the camera reference frame. As a result, depth information can be extracted from motion and measurement models through a recursive state estimation process. This section proposes motion, structure, and measurement models for the planar motion problem and investigates the application of extended Kalman filtering techniques to the estimation of motion and structural parameters.

4.3.1 System Geometry

Figure 4.17 shows a typical physical arrangement for this problem. The object of interest, again shown here as a block, is described by feature points, for example the block corners, in an object-centred reference frame, F_O . The x_O - y_O plane of F_O is defined to be parallel to the plane of motion in which case the axis of rotation is parallel to z_E . An earth fixed reference frame, F_E , is defined such that the x_E - y_E coordinate plane is the plane of motion. The position and orientation of the camera reference frame, F_C , is assumed to be known with respect to F_E .

As in Broida and Chellappa's approach, this investigation considers the case of unforced motion for which translational and rotational velocities are constant. Translational motion states, therefore, are the position and velocity of the origin of F_O with respect to F_E , while rotational states are defined as the yaw angle (see Section 2.2.3) and rotational velocity of F_O relative to F_E . In the following development, a stationary camera is assumed, however, the resulting estimation algorithm can be naturally extended to the case of a moving observer by virtue of the employment of an intermediate, earth-fixed frame. Moreover, parameterization of object motion with respect to F_E , as opposed to F_C , reduces the dimension of the state vector and hence simplifies the state estimation problem.

4.3.2 Motion and Structure Model

Parameterization of object motion relative to F_E is similar to that employed in Section 3.3, except three dimensional analysis now applies and object structure is included in the state vector. The origin of F_O lies on the plane of motion in

Figure 4.17: System geometry for planar motion analysis.

which case the position of F_O relative to F_E can be described with $[T_O^E](t) = [x(t), y(t), 0]^T$. The instantaneous orientation of F_O relative to F_E is described by the yaw angle $\psi(t)$ whose (constant) time derivative w defines the rotational velocity vector as $\mathbf{w} = [0, 0, w]^T$. A state vector for object motion in this case is

$$\mathbf{x}_m(t) \triangleq [x, \dot{x}, y, \dot{y}, \psi, w]^T. \quad (4.37)$$

Object structure is represented by a set of constant vectors in F_O ,

$$\mathcal{S} = \{\mathbf{r}_O^i; i = 1, 2, \dots, N_p\}, \quad (4.38)$$

where N_p is the number of feature points of interest. As discussed in Section 2.2.2, N_p is in general a non-decreasing, integer-valued function of time: as new feature points not already in $\mathcal{S}$ become of interest, they can be added to the structural model. However, due to feature point occlusion, only a subset $\mathcal{S}_v \subseteq \mathcal{S}$ may be observed in any given image. The occlusion problem has already been investigated for the geometry of Chapter 3 where structure was assumed to be known. Extensions of the following system and measurement models to the case of occlusion is deferred to Section 4.3.4.

If object structure is totally unknown, the object-centred reference frame must be rigidly fixed on the object—a problem that has already been discussed in the context of Broida and Chellappa's approach of Section 4.2. In the case of motion on a known plane without prior knowledge of structure, F_O must not only be fixed on the object, but must also be fixed in height with respect to the plane of motion. For the constant translational and rotational velocity case considered here, the origin of F_O lies at the intersection of the rotational axis and the plane of motion. In order to eliminate an arbitrary scale factor associated with estimation of object structure and translational motion, one feature point must be observed whose height above the plane of motion is known. That is, if one feature point is observed which, for example, is known to lie in the plane of motion, the z_O -coordinate of this feature point can be set to zero and eliminated from the estimation process. This practice provides essential depth information required for estimation of absolute position, translational motion, and structure. One angular degree of freedom about the rotational axis remains in

the specification of F_O relative to the object. Broida and Chellappa have treated this problem through absolute prior knowledge of the orientation of F_O relative to F_C (perfect alignment, for example). However, it is not clear whether this practice will yield a robust estimation scheme in the presence of system (plant) noise. An alternate approach is to assume, without loss of generality, that the observed feature point which lies at a known height above the plane of motion also lies in the (vertical) x_O - z_O plane. In the case where this special feature point lies in the plane of motion, both its x_O - and y_O -coordinates are zero and removed from the estimation process. Moreover, this feature point may be assumed to lie on the positive (or negative) x_O -axis. This problem is addressed more completely in Section 4.3.4 in the context of feature point occlusion. The development of the motion and structure models given below employs the full structure set S .

A state vector corresponding to object structure is defined as

$$\mathbf{x}_s \triangleq \begin{bmatrix} \mathbf{r}_O^1 \\ \mathbf{r}_O^2 \\ \vdots \\ \mathbf{r}_O^{N_p} \end{bmatrix}, \quad (4.39)$$

where each $\mathbf{r}_O^i$ is a column vector with components written as

$$\mathbf{r}_O^i = [r_{O,1}^i, r_{O,2}^i, r_{O,3}^i]^T. \quad (4.40)$$

The full state vector which describes object motion and structure is obtained through concatenation of (4.37) and (4.39) to write

$$\mathbf{x}(t) = \begin{bmatrix} \mathbf{x}_m(t) \\ \mathbf{x}_s \end{bmatrix}. \quad (4.41)$$

Under the assumption of constant translational and rotational velocity, a linear, discrete-time difference equation which describes time propagation of the state over a sample period T_k can be written as

$$\mathbf{x}(k+1) = \begin{bmatrix} \Phi(k) & \mathbf{0}_{6 \times 3N_p} \\ \mathbf{0}_{3N_p \times 6} & \mathbf{I}_{3N_p} \end{bmatrix} \mathbf{x}(k), \quad (4.42)$$

where $\mathbf{0}_{p,q}$ represents a $p \times q$ block of zeros, $\mathbf{I}_p$ is the $p \times p$ identity matrix, and

$$\mathbf{x}(k) = \begin{bmatrix} x(k) \\ \dot{x}(k) \\ y(k) \\ \dot{y}(k) \\ \psi(k) \\ \omega(k) \\ \mathbf{x}_s \end{bmatrix}, \quad \Phi(k) = \begin{bmatrix} 1 & T_k & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & T_k & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & T_k \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \quad (4.43)$$

Equation (4.42) is linear in the elements of the state vector of (4.41). Moreover, the overall state transition matrix for motion and structure is very sparse. These two properties of the motion-structure model can be exploited to improve execution speed of the time update stage for the discrete-time linear Kalman filter given in (2.56). In fact, since structural parameters are represented by constants, diffusion of confidence in estimates of these parameters between measurement events does not occur. This is demonstrated by writing the covariance matrix $\mathbf{P}(k|j)$ of (2.53) in block form as

$$\mathbf{P}(k|j) = \begin{bmatrix} \mathbf{P}_m(k|j) & \mathbf{P}_{ms}(k|j) \\ \mathbf{P}_{ms}^T(k|j) & \mathbf{P}_s(k|j) \end{bmatrix}, \quad (4.44)$$

where $\mathbf{P}_m$ is the motion parameter estimation error covariance, $\mathbf{P}_s$ is the structure parameter estimation error covariance, and $\mathbf{P}_{ms}$ is the motion-structure estimation error cross-covariance. In this case the Kalman filter time update of (2.56) can be written for a noise free system (plant) as

$$\hat{\mathbf{x}}(k+1|k) = \begin{bmatrix} \Phi(k)\mathbf{x}_m(k) \\ \mathbf{x}_s \end{bmatrix}, \text{ and} \quad (4.45)$$

$$\begin{aligned} \mathbf{P}(k+1|k) &= \begin{bmatrix} \Phi(k) & \mathbf{0}_{6 \times 3N_p} \\ \mathbf{0}_{3N_p \times 6} & \mathbf{I}_{3N_p} \end{bmatrix} \begin{bmatrix} \mathbf{P}_m(k|k) & \mathbf{P}_{ms}(k|k) \\ \mathbf{P}_{ms}^T(k|k) & \mathbf{P}_s(k|k) \end{bmatrix} \begin{bmatrix} \Phi(k) & \mathbf{0}_{6 \times 3N_p} \\ \mathbf{0}_{3N_p \times 6} & \mathbf{I}_{3N_p} \end{bmatrix}^T \\ &= \begin{bmatrix} \Phi(k)\mathbf{P}_m(k|k)\Phi^T(k) & \Phi(k)\mathbf{P}_{ms}(k|k) \\ \mathbf{P}_{ms}^T(k|k)\Phi^T(k) & \mathbf{P}_s(k|k) \end{bmatrix}. \end{aligned} \quad (4.46)$$

This representation shows that even though the structure covariance matrix $\mathbf{P}_s$ remains constant between observation events, the motion-structure cross-covariance

$\mathbf{P}_m$ propagates over time through the motion state transition matrix Φ . Equation (4.46) also demonstrates how time propagation of the state and its covariance can exploit the sparse nature of the state transition matrix in order to improve execution speed.

4.3.3 Measurement Model

The position of object feature points can be expressed with respect to F_E , as described in Section 2.2.2, through the rotation matrix

$$\mathbf{I}_E^O(\psi) = \begin{bmatrix} \cos \psi & -\sin \psi & 0 \\ \sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad (4.47)$$

and translation vector from the origin of F_E to that of F_O ,

$$[\mathbf{T}_O^E]_E = \begin{bmatrix} x \\ y \\ 0 \end{bmatrix}, \quad (4.48)$$

according to the relationship

$$\mathbf{r}_E^i = \mathbf{I}_E^O \mathbf{r}_O^i + [\mathbf{T}_O^E]_E. \quad (4.49)$$

Here, $[\mathbf{T}_O^E]_E$ is formed from the first and third elements of the motion state vector $\mathbf{x}_m$ defined in (4.37), and $\mathbf{I}_E^O$ is formed from ψ which is the fifth element of $\mathbf{x}_m$.

The position and orientation of the camera with respect to the earth-fixed frame is assumed to be known. This information is expressed in terms of a change-of-basis transformation from F_E to F_C , $\mathbf{I}_C^E$, and a translation from the origin of F_E to that of F_C written in F_C , $[\mathbf{T}_C^E]_C$. In this case, the position of the i th feature point on the object can be expressed with respect to F_C as

$$\mathbf{r}_C^i = \mathbf{I}_C^E \{ \mathbf{I}_E^O \mathbf{r}_O^i + [\mathbf{T}_O^E]_E \} - [\mathbf{T}_C^E]_C. \quad (4.50)$$

As in Section 2.2.2, $\mathbf{I}_C^E$ can be written as

$$\mathbf{I}_C^E = \begin{bmatrix} [i]_C & [j]_C & [k]_C \end{bmatrix}^T, \quad (4.51)$$

in which case the components of $\mathbf{r}_C^i = [r_{C,1}^i, r_{C,2}^i, r_{C,3}^i]^T$ are given by

$$\begin{aligned} r_{C,1}^i &= [\mathbf{i}_C]_E^T \{ \mathbf{I}_E^O \mathbf{r}_O^i + [\mathbf{T}_O^E]_E \} - [1, 0, 0] [\mathbf{T}_C^E]_C, \\ r_{C,2}^i &= [\mathbf{j}_C]_E^T \{ \mathbf{I}_E^O \mathbf{r}_O^i + [\mathbf{T}_O^E]_E \} - [0, 1, 0] [\mathbf{T}_C^E]_C, \text{ and} \\ r_{C,3}^i &= [\mathbf{k}_C]_E^T \{ \mathbf{I}_E^O \mathbf{r}_O^i + [\mathbf{T}_O^E]_E \} - [0, 0, 1] [\mathbf{T}_C^E]_C. \end{aligned} \quad (4.52)$$

The image position, $\mathbf{h}^i[\mathbf{x}(k)]$, of the i th feature point is then given by the perspective projection transformation of Equation (2.2),

$$\begin{aligned} \mathbf{h}^i[\mathbf{x}(k)] &= \begin{bmatrix} h_1^i[\mathbf{x}(k)] \\ h_2^i[\mathbf{x}(k)] \end{bmatrix} \\ &= \begin{bmatrix} L \frac{r_{C,2}^i}{r_{C,1}^i} \\ L \frac{r_{C,3}^i}{r_{C,1}^i} \end{bmatrix}, \end{aligned} \quad (4.53)$$

where L is the focal length. With N_p feature points of interest, and initially assuming a transparent object, the measurement model is given by

$$\mathbf{z}(k) = \begin{bmatrix} u_1(k) \\ v_1(k) \\ u_2(k) \\ v_2(k) \\ \vdots \\ u_{N_p} \\ v_{N_p} \end{bmatrix} = \mathbf{h}[\mathbf{x}(k)] + \mathbf{v}(k), \quad (4.54)$$

where

$$\mathbf{h}[\mathbf{x}(k)] = \mathbf{h} \begin{bmatrix} \mathbf{x}_m(k) \\ \mathbf{x}_s(k) \end{bmatrix} = \begin{bmatrix} \mathbf{h}^1[\mathbf{x}(k)] \\ \mathbf{h}^2[\mathbf{x}(k)] \\ \vdots \\ \mathbf{h}^{N_p}[\mathbf{x}(k)] \end{bmatrix}, \quad (4.55)$$

and $\mathbf{v}(k)$ models the measurement noise.

The Jacobian of $\mathbf{h}[\mathbf{x}]$, which is required in the EKF realization, is given by the $2N_p \times 6 + 3N_p$ matrix

$$\mathbf{H}(k) \triangleq \frac{\partial \mathbf{h}}{\partial \mathbf{x}} \quad (4.56)$$

$$= \begin{bmatrix} \frac{\partial \mathbf{h}^1}{\partial \mathbf{x}_m} \\ \vdots \\ \frac{\partial \mathbf{h}^{N_p}}{\partial \mathbf{x}_m} \end{bmatrix} \text{diag} \left\{ \frac{\partial \mathbf{h}^i}{\partial \mathbf{r}_O^i}; i = 1, \dots, N_p \right\}.$$

Assuming that $\mathbf{I}_E^O(\psi)$ and $\mathbf{r}_C^i$ have been computed as in (4.47) and (4.50), the block component matrices $\frac{\partial \mathbf{h}^i}{\partial \mathbf{x}_m} \in R^{2 \times 6}$ and $\frac{\partial \mathbf{h}^i}{\partial \mathbf{r}_O^i} \in R^{2 \times 3}$ in (4.56) can be represented as follows:

$$\frac{\partial \mathbf{h}^i}{\partial \mathbf{x}_m} = \begin{bmatrix} \frac{\partial h_1^i}{\partial x} & 0 & \frac{\partial h_1^i}{\partial y} & 0 & \frac{\partial h_1^i}{\partial \psi} & 0 \\ \frac{\partial h_2^i}{\partial x} & 0 & \frac{\partial h_2^i}{\partial y} & 0 & \frac{\partial h_2^i}{\partial \psi} & 0 \end{bmatrix}, \quad (4.57)$$

where

$$\begin{aligned} \frac{\partial h_1^i}{\partial x} &= L \left[\frac{r_{C,1}^i \alpha_{21} - r_{C,2}^i \alpha_{11}}{(r_{C,1}^i)^2} \right] \\ \frac{\partial h_1^i}{\partial y} &= L \left[\frac{r_{C,1}^i \alpha_{22} - r_{C,2}^i \alpha_{12}}{(r_{C,1}^i)^2} \right] \\ \frac{\partial h_1^i}{\partial \psi} &= -L \left[\frac{(\mathbf{r}_C^i)^T \mathbf{k}_E^i \mathbf{I}_C^E \mathbf{I}_E^O \mathbf{k}_E^i \mathbf{r}_O^i}{(r_{C,1}^i)^2} \right] \\ \frac{\partial h_2^i}{\partial x} &= L \left[\frac{r_{C,1}^i \alpha_{31} - r_{C,3}^i \alpha_{11}}{(r_{C,1}^i)^2} \right] \\ \frac{\partial h_2^i}{\partial y} &= L \left[\frac{r_{C,1}^i \alpha_{32} - r_{C,3}^i \alpha_{12}}{(r_{C,1}^i)^2} \right] \\ \frac{\partial h_2^i}{\partial \psi} &= L \left[\frac{(\mathbf{r}_C^i)^T \mathbf{j}_E^i \mathbf{I}_C^E \mathbf{I}_E^O \mathbf{k}_E^i \mathbf{r}_O^i}{(r_{C,1}^i)^2} \right] \end{aligned} \quad (4.58)$$

where $\mathbf{I}_C^E \triangleq [\alpha_{jk}]$ $j, k = 1, 2, 3$;

(4.59)

and

$$\frac{\partial \mathbf{h}^i}{\partial \mathbf{r}_O^i} = \begin{bmatrix} \frac{\partial h_1^i}{\partial r_O^i} \\ \frac{\partial h_2^i}{\partial r_O^i} \end{bmatrix} \quad (4.60)$$

where

$$\begin{aligned}\frac{\partial h_1^i}{\partial \mathbf{r}_O^i} &= -L \left[\frac{(\mathbf{r}_C^i)^T \mathbf{k}_E^\# \mathbf{I}_C^E \mathbf{I}_E^O}{(r_{C,1}^i)^2} \right] \\ \frac{\partial h_2^i}{\partial \mathbf{r}_O^i} &= L \left[\frac{(\mathbf{r}_C^i)^T \mathbf{j}_E^\# \mathbf{I}_C^E \mathbf{I}_E^O}{(r_{C,1}^i)^2} \right].\end{aligned}\quad (4.61)$$

All entries of $\mathbf{H}$ other than those specified above are zero. As in previous developments, $\mathbf{j}_E = [0, 1, 0]^T$, $\mathbf{k}_E = [0, 0, 1]^T$, and $\mathbf{j}_E^\#$ and $\mathbf{k}_E^\#$ are the matrix cross product operators associated with $\mathbf{j}_E$ and $\mathbf{k}_E$, respectively.

Neglecting occlusion and assuming object structure is unknown, a single image provides $2N_p$ equations in $6 + 3N_p - 2$ unknown variables since one feature point is assumed to lie on the $\mathbf{x}_O$ axis. Each additional feature point provides two more equations, but increases the dimension of the state vector ($\mathbf{x}_m$) by three unknowns. With four feature points, for example, each image provides 8 equations in 16 unknown variables. In this case, at least two image frames are required to obtain more equations than unknowns. Three of these variables (x, y, ψ) are time varying while all others are constant in this investigation.

4.3.4 Structure Estimation With Feature Point Occlusion

As the object rotates and translates, feature points naturally become occluded and may reappear at some future time depending on the object trajectory. Another form of occlusion occurs when the object becomes partially or completely hidden from view as it moves behind another object in the foreground of the scene. The effect of feature occlusion has already been investigated in Section 3.3 when structure of the object is known. Under known structure, the estimation algorithm proceeds as in the case of no occlusion, but the dimension of the measurement vector $\mathbf{z}(k)$ is time varying, i.e., $\mathbf{z}(k) \in R^{m(k)}$, where now $m(k) = 2N_v(k)$ and $N_v(k)$ is the number of visible feature points in the k th image.

When structure is unknown, the portion of the state vector corresponding to structure, $\mathbf{x}_s$, may contain unobservable states if formed from the complete struc-

ture set S . If a feature point is occluded over a short period of time, only slight degradation of performance should be expected. However, occlusion over a long time period should be treated by filter decoupling to remove estimation of structure associated with this point. Generally, the structure states of this feature point are removed from the state vector, and corresponding rows and columns are removed from the covariance matrix. If at some future time this feature is again recognized, its most recent structure estimates could be included in the state vector, and the covariance matrix would be augmented by relevant parameters, propagated in time according to (4.46), which measure the confidence in this previous estimate. This practice would result in a reconstruction of the state vector and covariance matrix with each sampled image. Further difficulties arise when the special feature point whose height above the plane is known becomes occluded from view. In this case, it may be possible to select another visible feature point for which structure estimation has the highest confidence, remove two coordinates of this second point from the state vector and proceed as before. Once confidence levels of any structural parameter estimate reach some threshold level, it may be removed from the state vector, but, if required, could still be employed in the estimation process. In this way, active selection of available feature points may lead to improved motion tracking performance and, in addition, emphasis could be shifted to structure estimation of new feature points in order to construct a more complete and meaningful model of the observed object.

For simplicity in the following investigations, the full state vector is employed throughout each trajectory regardless of which points are occluded. No attempt is made to devise and describe a complex recipe to adaptively build the state vector and covariance matrix at each measurement event. Although future investigations should consider this alternative, a "best" recipe may be somewhat application specific. Moreover, complete state estimation in the presence of occlusion will demonstrate the level of performance degradation and reduced accumulation of information that may be expected during the occlusion process.

4.3.5 Filter Initialization

The recursive estimation algorithm proposed in this section requires an initial estimate of the state, $\mathbf{x}(0|-1) \triangleq \mathbf{x}_0$ and characterization of the initial uncertainty, $\mathbf{P}(0|-1) \triangleq \mathbf{P}_0$, in this estimate. Broida and Chellappa have studied a batch approach [26, 27, 28, 31] which employs the Conjugate Gradient descent technique from the IMSL library. Simulation results in [28], for example, employ what are called "crude" initial estimates obtained from the first 10 image frames with 75 iterations of the batch technique. However, initial estimates used to generate the graphical results shown in [28] have not been given. When object motion is constrained to a known plane with one observed point at a known height from the plane, a simple initialization can provide suitable estimates for object position and orientation.

With central projection image formation geometry and focal length L , if N_v points are observed at image plane positions (u^i, v^i) , $i = 1, 2, \dots, N_v$, each vector $[L, u^i, v^i]^T_C$ lies along the ray from the centre of projection through the i th image point. Unit vectors along the i th ray are defined with respect to F_C as

$$\mathbf{u}_C^i \triangleq \frac{[L, u^i, v^i]^T_C}{\| [L, u^i, v^i]^T_C \|}. \quad (4.62)$$

The position of the i th feature of interest whose image appears at (u^i, v^i) is given by

$$\mathbf{r}_C^i = \lambda^i \mathbf{u}_C^i, \quad (4.63)$$

where each $\lambda^i > 0$. If this feature lies ℓ^i units above the plane of motion, then with $[\mathbf{T}_E^C]_E$ and $\mathbf{I}_E^C$ given,

$$\mathbf{r}_E^i = \lambda^i \mathbf{I}_E^C \mathbf{u}_C^i - [\mathbf{T}_E^C]_E, \quad (4.64)$$

where

$$\lambda^i = \frac{-\ell^i + [0, 0, 1][\mathbf{T}_E^C]_E}{[0, 0, 1]\mathbf{I}_E^C \mathbf{u}_C^i}. \quad (4.65)$$

In the case of known structure, the height of each feature point above the plane of motion is known from its z_O -component. For unknown structure, on the other hand, it may be the case that some expected mean height of the object could be employed for all feature points whose height is unknown. If $\mathbf{r}_E^i$ is computed with the above algorithm for each available feature point in the first image, the initial

position of the object can simply be approximated by the average of these vectors with the z_E -component set to zero:

$$[\dot{\mathbf{T}}_O^E]_E = \frac{1}{N_v} \sum_{i=1}^{N_v} \begin{bmatrix} r_{E,1}^i \\ r_{E,2}^i \\ 0 \end{bmatrix}. \quad (4.66)$$

If $\mathbf{r}_E^i$ corresponds to the special feature point whose height above the plane is assumed to be known and which is further assumed to lie on the positive x_O axis, the initial orientation ψ_O may be approximated by

$$[\dot{\mathbf{i}}_O]_E = \begin{bmatrix} \cos(\hat{\psi}_O) \\ \sin(\hat{\psi}_O) \\ 0 \end{bmatrix} = \frac{\mathbf{r}_E^i - [\dot{\mathbf{T}}_O^E]_E}{\|\mathbf{r}_E^i - [\dot{\mathbf{T}}_O^E]_E\|}. \quad (4.67)$$

Of course, the available feature points need not describe the extent of the object and, moreover, visible feature points lie towards one side of the object. As a result, a single frame may produce biased initialization. Depending on the available time and application, it may be beneficial to consider two or more frames for initialization in which case some information could be obtained concerning translational and rotational velocities. Further examination of filter initialization is given in the following section which presents simulation results for this technique.

4.3.6 Results and Performance With Simulated Imagery

The geometry of Figure 4.17 was used to generate sequences of image feature points with occlusion and correspondence information. All simulations of this section are shown with feature occlusion. Details of imagery generation with occlusion and other software employed in this study are included in Appendix A. The object is again described by the eight corners of a solid rectangular block with feature point labels as shown in Figure 4.4. As in Case 4.2-2 of the previous section, the block is 20 units in width (y_O direction) with a height (z_O) and depth (x_O) of 10 units. In all cases considered here, a pseudo random Gaussian noise sequence with standard deviation $\sigma_v = 0.02$ units was added to true feature point positions in the image.

This noise level again represents noise levels from 5% to 10% of the maximum size of the object image over the trajectory (when the object is closest to the camera). In all trajectories, the object moves away from the camera giving rise to increased noise levels in comparison to the object image size. The sampling period is taken as 0.1 seconds.

The camera position and orientation for the following investigations is defined by the translation

$$[\mathbf{T}_C^E]_E = [0, -30, -30]^T, \quad (4.68)$$

and transformation $\mathbf{I}_C^E$ formed from the Euler angles $\psi_C = 45^\circ$ (yaw or pan), $\theta_C = -45^\circ$ (pitch or tilt), and $\phi_C = 0$ (roll about optical axis) according to (2.12) and (2.13). These definitions were also used in Case 4.2-2 for Broida and Chellappa's method.

Industrial robots often work in a very controlled environment and on specific objects for which a great deal of prior information may be available concerning object structure. Motion parameter estimation under known structure is, therefore, an important case to consider. Filter performance for both the known and unknown structure cases have been investigated. The discrete-time extended Kalman filter, reviewed in Section 2.3.2, yields satisfactory results without local iterations when object structure is known, but demonstrates slow convergence in the unknown structure case. All simulation results shown for this technique employ local iterations with a maximum of 10 and terminating if the maximum element of the difference in successive iterates is less than 0.01. In most cases, only 5 to 7 iterations were required during initial observation events. As the filter accumulates information, off diagonal elements of the covariance matrix, which is initially a diagonal matrix, become nonzero leading to improved performance. In this case, only one or two iterations per observation event were required. It was noted, however, that when an occluded feature point suddenly becomes visible, the filter employed 4 or 5 iterations for the next few measurement updates and showed improved performance in comparison to the standard EKF implementation.

Simulation studies for this estimation technique include the following three cases:

Case 4.3-1 Object structure is assumed to be known. A comparison of Cramer Rao bounds for motion estimation is given in terms of the number of available feature points with occlusion. Sample and Monte Carlo simulation results are shown for motion estimation for a trajectory with low translational rates and relatively high rotational rates and is identical to that employed in Case 4.2-2.

Case 4.3-2 Sample and Monte Carlo simulation results are shown for motion and structure estimation with the low translational rate and high rotational rate trajectory employed in Case 4.3-1 and Case 4.2-2.

Case 4.3-3 Sample and Monte Carlo simulation results are shown for motion and structure estimation for a trajectory with higher translational rates and reduced rotational rates. In comparison to the trajectory of Case 4.3-2, feature points are occluded over longer periods of time due to the reduced angular velocity, and large object depth near the end of the run results in measurement noise levels on the order of 20% of the object image size. Almost twice as many observation events are required in this case to achieve the same accuracy as shown for Case 4.3-2, however, RMSE performance is more accurately predicted, in most states, by the Cramer Rao bounds.

This investigation concludes with a summary of results and a general discussion of the proposed technique for planar motion in comparison to the more general methods proposed by Broida and Chellappa.

Case 4.3-1: Cramer Rao bound investigation and simulation results for motion estimation under known structure. Four feature points common to all simulations of this section and simulations of Section 4.2 are, from Figure 4.4, p^1 , p^3 , p^5 , and p^7 . Of these four points, only p^1 and p^5 are subject to occlusion due to object motion. The structure of the object is described by the set

$$S = \left\{ \begin{bmatrix} 5 \\ 10 \\ 0 \end{bmatrix}, \begin{bmatrix} -5 \\ 10 \\ -10 \end{bmatrix}, \begin{bmatrix} -5 \\ -10 \\ 0 \end{bmatrix}, \begin{bmatrix} 5 \\ -10 \\ -10 \end{bmatrix} \right\}. \quad (4.69)$$

Table 4.3 lists the simulation parameters for this case. Figure 4.18 shows the

Table 4.3: Simulation parameters for Case 4.3-1.

$\mathbf{x}$	$\mathbf{x}(0)$	$\dot{\mathbf{x}}_0$	$\sigma(\dot{\mathbf{x}}_0)$	$T = 0.1 \text{ Sec.}$
x	30.0	34.49	5.0	units
$\dot{x}$	0.5	0.0	3.0	units/s
y	-5.0	-4.35	5.0	units
$\dot{y}$	1.0	0.0	3.0	units/s
ψ	1.0	1.25	0.3	radians
$\dot{\psi}$	-1.0	0.0	1.0	radians/s

true trajectory of image feature points, reproduced from Figure 4.11 for convenience, along with a feature occlusion map for this trajectory. A sample noisy image is shown in Figure 4.11 for additive measurement noise with standard deviation $\sigma_v = 0.02$ units. The sample period for this case is $T = 0.1$ seconds over a 10 second time interval. Feature point p^1 is visible in the first image frame, but then is immediately occluded for about two seconds. Feature point p^5 is occluded from 3 to 5 seconds and again from about 9.25 seconds to the end of the trajectory.

In Section 4.2, as in [28], four feature points were assumed to be available in each image without occlusion. The number of feature points determines the dimension of the observation vector $\mathbf{z}(k)$ and, in particular, the dimension of the matrix inversion problem in the computation of the gain matrix. Four feature points may at first seem to be a rather small number, particularly in the case of occlusion for which the estimation process may rely on fewer feature points over a significant portion of the trajectory. At perhaps the opposite extreme, Wu *et. al.* [32] show results for 14 feature points without occlusion and with known structure. Cramer Rao bound analysis shown in Case 3.3-3, however, demonstrated the decreasing returns in performance improvements as the number of feature points was increased for the particular geometry of Chapter 3.

Figure 4.19 shows Cramer Rao bounds in the absence of prior information ($\mathbf{P}_0^{-1} = \mathbf{0}$) for this trajectory as the number of feature points is increased from 4 to 8. The upper solid trace in each plot represents covariance bounds when the standard four

Table 4.4: Sample initialization statistics for Case 4.3-1.

Parameter	True Value	Mean Error	Error STD	
x	30.0	-0.2856	1.663	units
y	-5.0	-0.5745	1.705	units
ψ	1.0	0.0015	0.205	radians

feature points p^1 , p^3 , p^5 , and p^7 are used. The dashed, dotted, dash-dotted, and lower solid traces result, respectively, from inclusion of feature points p^2 , p^4 , p^6 , and p^8 . An immediate observation is that only slight improvements in performance bounds result when more than four feature points are considered. The feature occlusion map shown in Figure 4.18 should be compared to the results shown in Figure 4.19. For example, with the standard four feature points (upper solid trace), the appearance of p^1 at $t = 2$ seconds coincides with a sudden, but small decrease in y -position and angular position lower bounds, and the occlusion of p^5 during the period $2 < t < 5$ seconds corresponds in time to the slight increase in x -position and angular position lower bounds. These plots also demonstrate that an optimal unbiased estimation scheme, under measurement noise levels of standard deviation $\sigma_v = 0.02$ units and known structure, can only estimate x - and y -position to within a one-sigma error of about 0.15 units and angular position to within 0.01 radians (0.57°) over the 10 second trajectory. Position and orientation lower bounds are decreasing only slightly at the end of the trajectory, while translational and angular velocity performance bounds are quite low and still decreasing at a significant rate at 10 seconds.

The sample initialization, $\hat{x}_0$, shown in Table 4.3 was computed through the single frame initialization algorithm of Section 4.3.5 with measurement errors of $3\sigma_v$. Larger measurement errors are assumed for initialization since feedback from motion estimation would not be available to aid the feature detection process. In this trajectory, all four feature points are visible in the first image, however, p^1 immediately becomes occluded for 2 seconds. Statistics compiled from 60 initialization computations with this measurement noise level are shown in Table 4.4. For this trajectory, initial position estimates are slightly biased with one-sigma errors on the

order of 2 units. Initial orientation estimates, on the other hand, show only small bias, but have one-sigma errors of 0.2 radians (11.5°).

Figure 4.20 shows sample results obtained with filter initialization as shown in Table 4.3. The initial covariance matrix was estimated as

$$\mathbf{P}_0 = \text{diag}[\sigma^2(\dot{\mathbf{x}}_0)]. \quad (4.70)$$

Position and orientation errors remain noisy throughout the run, but demonstrate satisfactory convergence after 5 seconds (50 image samples). Good convergence is demonstrated for translational and rotational velocity estimation in approximately 2.5 seconds. Figure 4.21 provides sample results when the filter is initialized with an all-zero state vector. These results are quite satisfactory and demonstrate that information available in the measurement process together with perfect knowledge of structure is sufficient to yield convergence in the presence of large initialization errors. However, filter initialization, at least in position and orientation, is required in the unknown structure case in order to achieve satisfactory results.

Figure 4.22 shows root-mean-squared-errors (RMSE), Cramer-Rao lower bounds (CRLB), and mean error (ME) results for a 60 run Monte Carlo simulation with filter initialization taken in accordance with (2.92) as $\hat{\mathbf{x}}(0|-1) \sim N(\mathbf{x}(0), \mathbf{P}_0)$, where $\mathbf{P}_0$ is defined in (4.70). There were no cases in which the filter failed to converge to within a reasonable interval of the true motion parameters. Filter performance in all states is very accurately predicted by the Cramer Rao bounds. Only slight bias is seen in angular position estimation over the first 5 seconds (50 samples). During occlusion of p^5 from 3 to 5 seconds and also during the second occlusion of p^1 from approximately 6 to 8 seconds, angular position RMS errors tend to increase which results in increased errors in the convergence index $J(k)$ for the rotational operator. However, final errors in $J(k)$ are approximately 0.5%.

Figure 4.18: True image sequence and occlusion map for Cases 4.3-1 and 2.

Figure 4.19: Cramer Rao bound variation with number of feature points under known structure and occlusion. All plots show upper — four, --- five, six, - . - . - seven, and lower — eight feature points.

Figure 4.20: Sample results for Case 4.3-1 with single frame initialization.

Figure 4.21: Sample results for Case 4.3-1 with all zero initialization.

Figure 4.22: Monte Carlo simulation results for Case 4.3-1. All plots show — RMSE, --- CRLB, and ME.

Table 4.5: Simulation parameters for Case 4.3-2.

$\mathbf{x}$	$\mathbf{x}(0)$	$\hat{\mathbf{x}}_0$	$\sigma(\hat{\mathbf{x}}_0)$	$T = 0.1$ Sec.
r	30.0	26.9	5.0	units
$\dot{x}$	0.5	0.0	3.0	units/s
y	-5.0	-6.92	5.0	units
$\dot{y}$	1.0	0.0	3.0	units/s
ψ	2.1071	1.78	0.35	radians
w	-1.0	0.0	1.0	radians/s
$\mathbf{r}_O^i$	S	0.0	5.0	units

Case 4.3-2: Motion and structure estimation for high rotation, low translation trajectory. The structure of the object is again described by the four feature points p^1 , p^3 , p^5 , and p^7 . Feature point p^5 is known to lie in the plane of motion and is further assumed to lie on the negative $\mathbf{x}_O$ axis. In this case, the true structure set describes the same feature points as in Case 4.3-1, but is rotated about $\mathbf{z}_O$ in order to coincide with structural parameters that the filter is designed to estimate. Rotation of F_O about $\mathbf{z}_O$ through an angle $\gamma \triangleq \tan^{-1}(2) = 1.107$ radians yields the structure set

$$\begin{aligned}
 S &= \{\mathbf{r}_O^1, \mathbf{r}_O^3, \mathbf{r}_O^5, \mathbf{r}_O^7\} \\
 &= \left\{ \begin{bmatrix} 11.18 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 6.708 \\ 8.944 \\ -10 \end{bmatrix}, \begin{bmatrix} -11.18 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} -6.708 \\ -8.944 \\ -10 \end{bmatrix} \right\}. \quad (4.71)
 \end{aligned}$$

Figure 4.18 has already demonstrated the true trajectory of image feature points and occlusion map for this trajectory. A sample noisy image was shown in Figure 4.11 for additive measurement noise with standard deviation $\sigma_v = 0.02$ units. The sample period for this case is $T = 0.1$ seconds.

Table 4.5 lists the simulation parameters for this case. The sample initialization, $\hat{\mathbf{x}}_0$, shown in Table 4.5 was computed through the single frame initialization algorithm with measurement errors of $3\sigma_v$. Recall that this algorithm allows one to specify a mean expected height of the object, which in this case was selected as 4

units, and then computes an estimate of object position assuming all feature points lie at this height above the plane of motion. Initialization statistics for this technique are not shown since they are almost identical to those provided in Table 4.4 of Case 4.3-1.

Figure 4.23 shows sample results obtained with filter initialization as shown in Table 4.5. Position and translational velocity errors are initially quite noisy, but demonstrate satisfactory convergence after 5 seconds (50 image samples). Good convergence is demonstrated for rotational velocity estimation in approximately 2.5 seconds. Angular position errors, however, remain slightly biased throughout the run. Structure estimation of non-occluded points p^3 and p^7 also shows convergence by 5 seconds, but with slight bias in the x_0 and z_0 -coordinates of p^7 . Structure errors for p^1 remain large during occlusion, but also converge after 5 seconds, as does the x_0 -coordinate error for p^5 .

Figures 4.24, 4.25, and 4.26, show root-mean-squared-errors (RMSE), Cramer-Rao lower bounds (CRLB), and mean error (ME) results for a 60 run Monte Carlo simulation with filter initialization taken in accordance with (2.92) as $\hat{x}(0|-1) \sim N(x(0), P_0)$, where P_0 is defined in (4.70). There were no runs for which the filter failed to converge to within a reasonable interval of the true motion and structural parameters.

Figure 4.24 shows results for motion parameter estimation and the convergence index $J(k)$ for the rotational operator. Estimation of x -position and velocity, and angular position tend to be biased over the first 5 seconds, but after p^5 again becomes visible at 5 seconds, the RMSE for these parameters approach and follow the Cramer Rao bounds. RMS errors for y -position and velocity are also initially biased, but approach and follow the CRLB after p^1 becomes visible at 2 seconds. Good performance is shown in the estimation of angular velocity. The convergence index $J(k)$ shows a final error of approximately 1%. A comparison of these results with those of Case 4.3-1 in Figure 4.22 demonstrates the degradation in motion parameter estimation that might be expected while structural parameters are being estimated.

Figure 4.25 shows the results for structure estimation of points p^3 and p^7 which

are not subject to occlusion. Estimation of the x_O - and y_O -coordinates of p^3 are biased over the first 5 seconds, and estimates of the x_O -coordinate of p^7 remain biased over the run with an error of approximately 7% at the end of the run. Similar behavior is shown in Figure 4.15 for Broida and Chellappa's technique. However, in Figure 4.25, good performance is shown in the estimation of the height of feature points (z_O) and in the y_O -coordinate of p^7 .

Figure 4.26 shows results for structure estimation of point p^1 and also the x_O -coordinate of p^5 . For this trajectory, p^1 is visible only in the first image sample and then is occluded for 2 seconds. During this occlusion, state estimation is based on only three feature points, and significant errors remain in estimates of p^1 structure, which may be responsible for bias seen in other states over the first 2 seconds. The presence of this point in images sampled after $t = 2$ seconds results in a rapid decrease in RMS errors as estimation performance approaches the Cramer Rao bounds. Estimation of the x_O -coordinate of p^5 is also somewhat biased particularly just after p^1 becomes visible and during the time interval $3 < t < 5$ seconds when p^5 itself is occluded. RMS errors, however, approach the CRLB towards the end of the run with a final error of approximately 2.5%.

Figure 4.23: Sample results for Case 4.3-2.

Figure 4.24: Motion parameter Monte Carlo simulation results for Case 4.3-2. All plots show — RMSE, - - - CRLB, and ME.

Figure 4.25: Structure parameter Monte Carlo simulation results for feature points p^3 and p^7 for Case 4.3-2. All plots show — RMSE, - - - CRLB, and ME.

Figure 4.26: Structure parameter Monte Carlo simulation results for feature points p^1 and p^5 for Case 4.3-2. All plots show — RMSE, - - - CRLB, and ME.

Table 4.6: Simulation parameters for Case 4.3-3.

$\mathbf{x}$	$\mathbf{x}(0)$	$\dot{\mathbf{x}}_0$	$\sigma(\dot{\mathbf{x}}_0)$	$T = 0.1 \text{ Sec.}$
x	30.0	27.72	5.0	units
$\dot{x}$	2.5	0.0	3.0	units/s
y	-25.0	-26.05	5.0	units
$\dot{y}$	5.0	0.0	3.0	units/s
ψ	2.1071	2.02	0.35	radians
w	-0.5	0.0	1.0	radians/s
r_O^i	S	0.0	5.0	units

Case 4.3-3: Motion and structure estimation for high translational velocity and reduced angular velocity trajectory. The structure of the object is again described by four feature points p^1 , p^3 , p^5 , and p^7 defined by the structure set given in (4.71). Feature point p^5 is known to lie in the plane of motion and is further assumed to lie on the negative x_O axis.

Table 4.6 lists the simulation parameters for this case. Figure 4.27 shows the true image feature point trajectories and a sample noisy image for additive measurement noise with standard deviation $\sigma_v = 0.02$ units. The true initial depth of the object is approximately 40 units with a depth velocity (in the x_O direction) of 3.75 units/s. The initial object image size is from 0.25 to 0.5 units. In this case the measurement noise levels of 0.02 units represents quantities on the order of 5% to 10% of the object image size when the object is closest to the camera. A 20 second time interval is employed with a sample period of $T = 0.1$ seconds. The depth of the object at the end of the 20 second trajectory is 113 units for which measurement noise levels are 11% to 23% of the object image size. A feature point occlusion map is shown in Figure 4.28. In this case, feature point p^1 is visible in the first 9 image samples or almost 1 second before being occluded for 4 seconds. Again, the z_O -component of p^5 is assumed to be known as zero. This point is occluded during the time periods from 6 to 10 seconds and from about 19 seconds to the end of the trajectory.

The sample initialization, $\dot{\mathbf{x}}_0$, shown in Table 4.6 was computed through the sin-

gle frame initialization algorithm of Section 4.3.5 with measurement errors of $3\sigma_v$. The mean expected height of the object was again specified as 4 units. Figure 4.29 shows sample results obtained with filter initialization as shown in Table 4.6. Position and translational velocity errors are initially noisy, but demonstrate satisfactory convergence after 10 seconds (100 image samples) and 5 seconds (50 samples), respectively. Good convergence is demonstrated for rotational velocity estimation in approximately 2 seconds. Angular position errors, however, again remain slightly biased throughout the run. Structure estimation of non-occluded points p^3 and p^7 also show convergence by 7 seconds, except in the x_O -coordinate of p^7 which has a final error of 1 unit (14%). Structure errors for p^1 remain large during occlusion, but also converge after 7 seconds, as does the x_O -coordinate error for p^5 .

Figures 4.30, 4.31, and 4.32, show root-mean-squared-errors (RMSE), Cramer-Rao lower bounds (CRLB), and mean error (ME) results for a 60 run Monte Carlo simulation. There were no runs for which the filter failed to converge to within a reasonable interval of the true motion and structure parameters. Figure 4.30 shows results for motion parameter estimation and the convergence index $J(k)$ for the rotational operator. Estimation of x -position and angular position tend to be biased over the first 10 seconds. Angular position RMSE in fact remains above the CRLB throughout the trajectory. After p^5 again becomes visible at 10 seconds, the RMSE for x -position approaches and follows the Cramer Rao bounds. RMS errors for y -position and velocity are also initially biased, but approach and follow the CRLB after p^1 becomes visible just after 4 seconds. Good performance is shown in the estimation of angular velocity. The convergence index $J(k)$ shows a final error, as in Case 4.3-2, of approximately 3%, but here twice as many observation events have been processed.

Figure 4.31 shows the results for structure estimation of points p^3 and p^7 which are not subject to occlusion. These results are in fact very similar to those for Case 4.3-2 in that satisfactory RMSE performance is demonstrated after 5 seconds. Estimates of the x_O -coordinate of p^7 again remain slightly biased over the run with a final error of approximately 7%. Figure 4.32 shows results for structure estimation of point p^1 and the x_O -coordinate of p^5 . For this trajectory, p^1 is visible in the

first 9 image samples and then is occluded for almost 4 seconds. As in Case 4.3-2, state estimation during this occlusion is based on only three feature points. However, errors in estimates of p^1 structure during this occlusion are lower than those in Case 4.3-2, due to the larger number of observations of this point prior to occlusion. These smaller errors may be responsible for the slightly reduced bias, in comparison to the results of Case 4.3-2, in other states over the initial portion of the trajectory. The presence of p^1 in images sampled after $t = 4$ seconds results in a rapid decrease in RMS errors as estimation performance approach the Cramer Rao bounds. Estimation of the x_0 -coordinate of p^5 is also somewhat biased particularly just after p^1 becomes visible and during the time interval $6 < t < 10$ seconds when p^5 itself is occluded. RMS errors, however, approach the CRLB towards the end of the run with a final error of approximately 2%.

Figure 4.27: True image sequence and sample noisy image for Case 4.3-3.

Figure 4.28: Occlusion map for trajectory of Case 4.3-3.

Figure 4.29: Sample results for Case 4.3-3.

Figure 4.30: Motion parameter Monte Carlo simulation results for Case 4.3-3. All plots show — RMSE, - - - CRLB, and ME.

Figure 4.31: Structure parameter Monte Carlo simulation results for feature points p^3 and p^7 for Case 4.3-3. All plots show — RMSE, - - - CRLB, and ME.

Figure 4.32: Structure parameter Monte Carlo simulation results for feature points p^1 and p^5 for Case 4.3-3. All plots show — RMSE, - - - CRLB, and ME.

4.3.7 Discussion

A fundamental assumption employed in applications of this technique to both motion and structure estimation is that the height of one feature point above the plane of motion is known, or can be approximated. Note that this assumption is quite distinct from one which assumes instantaneous depth information of an observed (or unobserved) point on (or relative to) the object is known *a priori* such as that required to obtain absolute position and structure using Broida and Chellappa's approach. The later assumption fixes this point to lie in planes which are parallel to the image plane and must be known (or estimated) at each observation event, while the former, which relies on a planar motion assumption, fixes this point to lie in a plane parallel to the known plane of motion for all observation events. If a feature point with known height above the plane of motion is not available and the height of the imaging system above the plane of motion is much larger than the expected height of the object, such as in aerial surveillance systems for example, a reasonable initial assumption may be that all feature points lie in a plane at some mean expected height from and parallel to the plane of motion. In other applications, it may be possible to select an observed feature point which appears to lie on the "bottom" of the object. Such approximations, however, would be highly application specific.

The initial performance of the proposed technique is not only dependent on parameter estimates and the covariance matrices, but also on the occlusion process. Performance degradation during feature point occlusion, however, should be expected from most motion recovery techniques which are formulated through a feature-based approach. Unfortunately, performance evaluations in the presence of occlusion are not generally available in the literature.

In all simulations shown here, the initial covariance matrix was chosen as diagonal. The inverse of the covariance matrix provides a measure of the information contained in the filter (see Section 2.4 and [44]). The initial diagonal matrix will not, in general, correctly model correlated state variables. Due to the sparse nature of the state transition matrix and the linearized observation matrix $\mathbf{H}$, a large number of measurement events are required before off diagonal elements of the covariance

matrix become non-zero. This "filling in" of the covariance matrix is also inhibited by the occlusion process which results in zero columns in the observation matrix $\mathbf{H}$ if states corresponding to structure of the occluded point are not removed. As a result, diagonal initialization of the covariance matrix may require a large number of time and measurement updates before correlation between state variables is estimated by the filter. This fact, coupled with the occlusion process, may be largely responsible for initial bias shown in results for motion and structure estimation. Batch techniques, such as those employed by Broida and Chellappa, may provide a better approximation for the initial covariance matrix and lead to improved performance at the expense of increased computational complexity during the initialization stage.

A direct performance comparison between the proposed technique for planar motion and Broida and Chellappa's method may not be appropriate since the later is more general than the former. Moreover, motion parameters are expressed with respect to different reference systems, results for the proposed technique are shown with feature point occlusion which Broida and Chellappa do not consider, and object depth is not estimated in Broida and Chellappa's method. In essence the task which the proposed estimation process must perform is simplified through the planar motion assumption, but hindered by the occlusion process and the requirement for depth estimation, while that associated with Broida and Chellappa's technique is simplified through the assumptions of available depth information and transparent objects, but hindered by the more general and nonlinear motion model and larger dimension of the state estimation problem.

A qualitative comparison of results of Case 4.3-2 with those of Case 4.2-2 indicates similar performance in the estimation of position, translational velocity, and structure. Estimation errors for these states tend to converge, or at least follow Cramer Rao bounds, in the second half of the trajectories. For an angular velocity of magnitude 1 rad/s these states require approximately 5 seconds (50 image samples) to demonstrate convergence. When the rotational velocity is reduced to 0.5 rad/s, however, convergence rates appear to be reduced by a factor of 2 in that 10 seconds (100 samples) are required for convergence. This slower convergence rate with reduced angular velocity was also observed for Broida and Chellappa's method.

In all cases, estimation of object orientation is slow to demonstrate convergence. In Broida and Chellappa's method, quaternion errors were oscillatory with decreasing amplitude, but RMS errors remained significantly above Cramer Rao bounds throughout the trajectory. Estimating a quaternion involves the recovery of a scaled trigonometric function of half the angle relating the two reference frames. The error between the true quaternion and that estimated by the filter is computed as the difference of two scaled and time shifted sinusoids. In particular, the period of oscillation shown in mean quaternion errors appears to be approximately twice the period of object rotation. In the yaw angle formulation for the planar motion case, the filter is required to estimate the angle of rotation $\psi(t)$ directly instead of a trigonometric function of this angle. Figures 4.33 and 4.34 show a performance comparison of the convergence index $J(k)$ defined in (2.44) for rotational operators $\mathbf{I}_C^Q$ formed from quaternion and yaw angle estimation for the two planar motion trajectories considered in this section. Figure 4.33 shows the RMS error of $J(k)$ for Case 4.2-2 of Broida and Chellappa's technique in comparison to that of Case 4.3-2 for the proposed method with planar motion. Recall that Broida and Chellappa assume the initial orientation is known exactly ($J(0) = 0$), whereas the proposed technique is provided with only finite information about the initial orientation, observes four feature points in the first image and only three points for the next 2 seconds, and is in fact unobservable during the period $3 < t < 5$ seconds since the special feature point p^5 is occluded. It is somewhat surprising, therefore, to see a similarity in the general shape of these curves, particularly in that both methods reach almost the same level of error before uncertainty is reduced. However, this similarity between Broida and Chellappa's method and the proposed technique is not present in Figure 4.34 for the higher translational velocity with reduced angular velocity trajectory of Case 4.3-3.

Figure 4.33: Comparison of rotation operator convergence index RMSE for Case 4.2-2 and Case 4.3-2. Plot shows — J -RMSE for Broida and Chellappa's algorithm, - - - J -RMSE for proposed planar motion technique.

Figure 4.34: Comparison of rotation operator convergence index RMSE for Case 4.3-3. Plot shows — J -RMSE for Broida and Chellappa's algorithm, - - - J -RMSE for proposed planar motion technique.

4.4 Chapter Summary

This chapter has reviewed the development of Broida and Chellappa's technique of [28] and has demonstrated simulation results for this method when object motion is constrained to lie on a plane. In general, satisfactory performance was demonstrated for object trajectories considered here, however, this approach is inappropriate for applications which require absolute position, translational motion, and structure of the observed object. In this case, depth information must be provided by an external source. Because such information would most likely be contaminated with measurement noise, a more realistic approach may be to include depth in motion and measurement models and remove the normalization of translational motion and structural parameters.

Motion, structure, and measurement models have been proposed for a particular class of motion analysis problems in which objects of interest are constrained to lie on a known planar surface. In addition to the planar motion constraint, complete structure can be recovered if one observed feature point lies at a known height above (or on) the plane of motion. Recursive state estimation, based on an iterated extended Kalman filter, has been applied to refine motion and structure parameter estimates over an arbitrary number of image frames. This investigation has demonstrated filter performance that might be expected for the case of constant translational and rotational velocity with feature point occlusion.

Position, translational motion, and structure of an object cannot be recovered from measurements of feature positions in monocular images without prior information through which depth information can be reconstructed. This is an immediate consequence of the projection of R^3 onto R^2 , a singular transformation, defined by the perspective projection image formation model. The literature review of Section 1.2 demonstrated the wide range of assumptions which have been employed to recover absolute information from monocular image sequences. Techniques examined in this chapter also depend on particular sets of assumptions and, therefore, can only be directly applied if these assumptions are met.

This chapter has focused on motion models and investigated filter performance

for constant translational and rotational velocity. It has been noted that neither of these techniques are generally applicable to the case of pure translational motion. If it is known *a priori* that motion is translational only, estimation of angular velocity can be removed, and one feature point can be selected as the origin of the object-centred frame and hence has known structure. In general, methods examined in this chapter can be applied provided an initialization scheme is available which will detect the presence or absence of rotational motion. However, good performance can only be expected if angular velocity is exactly zero, or is relatively high; object trajectories with "small" angular rates will most likely lead to serious degradation in position and structure estimation. This problem is again discussed in Chapter 5 which suggests possible approaches for further investigation.

Chapter 5

Conclusions and Proposals For Future Work

Application of sequential state estimation techniques such as the Kalman filter to machine vision problems emphasizes temporal continuity conditions for image sequence interpretation. Temporal continuity is described in the form of differential or discrete-time difference equations. Originally, Kalman [43] exploited linear dynamical models to formulate recursive least mean square estimation of state variables given noise corrupted measurements to yield estimates of the entire set of state variables, even though state variables are not measured directly. More recently, this famous principle has been extended to nonlinear problems, but usually results in suboptimal state estimation. However, application of this extended technique to dynamic vision problems circumvents the nonunique inversion of perspective projection image formation models by maintaining an internal representation, in the form of state variables, of the process being observed.

The recursive state estimation algorithms studied here are feature-based approaches which, in contrast to methods which focus on batch processing techniques of a fixed number of frames, refine estimates of motion and structural parameters over an arbitrary number of image frames and do not require storage of past image data. Moreover, time derivatives are estimated through smoothing numerical operations (integration) of prediction errors rather than discrete difference approximations commonly employed in estimations of time and spatial derivatives of discrete image

intensity functions in an image flow approach.

The work of Dickmanns and his colleagues [37]-[42] represents one of the primary motivations for the thesis work. This is one of the very few groups who have published results of complex experiments on real-time vision-based control through Kalman filtering. Their approach has evolved over the last decade from an initial specification of MIMD image sequence processing hardware to specialized applications in autonomous vehicle control. Published experimental results have demonstrated that a Kalman filtering approach is feasible in real-time applications provided appropriate hardware is available; control cycle times of 0.1 seconds have been reported [42] for autonomous vehicle control with a small set of commonly available microprocessors (INTEL 8086,80286). Broida and Chellappa, on the other hand, have considered a more general class of motion analysis problems and also employ a Kalman filtering technique. In addition, this method is not specific to a particular application and is generally accepted by virtue of numerous publications in major journals [24]-[28].

The theoretical review of Chapter 2 first introduced the perspective projection image formation model and a convenient notation for coordinate transformations which is generally applicable to analysis with multiple reference frames. An operational approach, characteristic of most engineering treatments of the subject, was taken in a review of linear and extended, discrete and continuous Kalman filters with local iterations. Approximate Cramer-Rao bounds were developed for the non-linear discrete time state estimation problem and provide a means for quantitative performance assessments of estimation algorithms. Computation of performance bounds follows essentially the same algorithm as that used to propagate the covariance matrix in the extended Kalman filter formulation, except that linearization of the system is taken about a true state trajectory.

The major focus of this investigation has been on motion and structure estimation under a planar motion constraint. Motion, structure, and measurement models have been proposed for the case in which objects of interest are constrained to lie, at least locally in space and time, on a known planar surface. A degenerate case in which the plane of motion contains the optical axis was examined in Chapter 3 with

reference to initial investigations conducted by Broida and Chellappa [24]. In this case, knowledge of the plane of motion does not yield essential depth information required for absolute position, translational motion, and structure estimation with a monocular imaging system. However, most applications which could exploit a planar motion constraint would be expected to observe the motion process at some arbitrary angle from a position above the known plane. This case was the major focus of Chapter 4. Broida and Chellappa's general approach for monocular imaging systems [28] was reviewed and applied to the planar motion problem. Simulation results for their method were compared to that obtained from the proposed technique which was developed specifically for the case of planar motion.

Evaluation of Broida and Chellappa's method and the proposed technique for planar motion has been conducted through Monte Carlo simulations with comparison of RMS errors to approximate Cramer-Rao bounds. Feature detection and correspondence problems, and prediction error feedback from the state estimation process have not been included in this investigation. One limitation of analysis based on simulated imagery is that the assumed central projection image formation model is exact; that is, the measurement model is based on the central projection transformation, and simulated imagery is generated precisely from this model. This approach, which is common to most simulation studies of image analysis, is only valid if distortion effects in the imaging system are small in comparison to measurement noise levels. A second limitation of this analysis lies in the use of noise-free dynamic models. Noise processes which drive dynamical models, however, are often extremely application specific. Investigations and performance evaluations in the absence of system noise places emphasis on the nonlinear measurement model and measurement noise levels, and allows direct comparison of results to approximate Cramer-Rao bounds which are generally available in non-linear problems only for a noise-free system.

Chapter 3 investigated the case in which the optical axis and central scan line of the imaging system lie in the plane of motion. Broida and Chellappa's approach to this problem in [24] demonstrated satisfactory performance provided object rotational rates are relatively high. As rotational rates decrease, even at $30^\circ/\text{s}$, motion

and structure estimation accuracy is seriously degraded. Moreover, the structural model employed in this approach does not fully exploit the rigid object assumption. Under an assumption of known structure, a slightly different motion model was proposed for the planar motion case. Simulation results for known structure demonstrated filter performance with respect to the number of feature points available and feature point occlusion. Satisfactory performance can be obtained with two feature points, however, convergence in some motion states is rather slow. If three feature points are available, filter performance is greatly improved in that the estimation root-mean-square-errors follow Cramer-Rao bounds. Increasing the computational burden by considering more than three feature points, however, yields decreasing returns in terms of performance bounds. This investigation also demonstrated that the recursive structure of the extended Kalman filter may provide natural solutions in the presence of occlusion in that the dimension of the measurement sequence need not remain constant.

The development of Broida and Chellappa's technique of [28] was reviewed in Chapter 4. Simulation results for this method were generated for the case in which object motion is constrained to lie on a plane. In general the performance of this approach for the two cases considered is satisfactory. Although some bias is seen during initial iterations of the filter, normalized position, velocity, and structure estimates demonstrate acceptable convergence. Quaternion estimation errors, however, show oscillatory behavior of decreasing amplitude and may require a large number of image samples to attain a high confidence level. Angular velocity estimation showed a constant bias, especially in the x_C - and z_C -components.

Linear motion and structure models, and a nonlinear measurement model were proposed in Chapter 4 for the case in which objects of interest are constrained to lie on a known planar surface. Recursive state estimation, based on an iterated extended Kalman filter, was applied to refine motion and structure parameter estimates over an arbitrary number of image frames. This investigation has demonstrated filter performance that might be expected for the case of constant translational and rotational velocity with feature point occlusion. Simulation results for motion estimation under known structure are very good in that RMS errors im-

3

of/de

3

MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS
STANDARD REFERENCE MATERIAL 1010a
(ANSI and ISO TEST CHART No. 2)

mediately attain and follow Cramer Rao bounds with negligible estimation bias. When structure is unknown, some estimation bias results during initial iterations particularly in the presence of occlusion. However, even with occlusion, root-mean-square errors of absolute position, translational motion, and structural parameters approach and follow Cramer Rao bounds during the later portion of simulation runs. Estimation of angular velocity is very good in either the known or unknown structure case. Angular position estimates with unknown structure demonstrate somewhat slow convergence, but improved convergence of the rotational operator has been demonstrated for the proposed technique in comparison to that shown for Broida and Chellappa's approach. This analysis also demonstrated that only slight improvements in performance bounds result when more than four feature points are considered.

For some initial state estimate errors, the filters examined in Chapter 3 had a tendency to converge to motion and structural parameters different from those employed to generate the imagery. This phenomenon has also been noted by Broida [31] and Broida and Chellappa [24]. In [31] it is suggested that different sets of motion and structural parameters can lead to essentially the same image sequence, so failure to converge to the "true" parameters does not necessarily imply filter divergence in the usual sense. Instead, there appears to be at least two solutions in state space to which the filters may converge, particularly for the specialized geometry considered in Chapter 3. A demonstration of the ambiguity of rotational motion when observed through the perspective projection transformation was provided in order to clarify this behavior of the state estimation algorithms. Failure to converge to within a reasonable interval of true motion and structural parameters did not occur for filters developed and analyzed in Chapter 4. However, Broida [31] has noted convergence to alternate sets of parameters when their general method is applied to monocular image sequences without feature point occlusion. Further analysis is required in order to establish uniqueness properties of the proposed planar motion state estimation algorithm.

In order to eliminate an arbitrary scale factor associated with the estimation of position, translational motion, and structure, Broida and Chellappa have refer-

enced these states to the image plane through normalization by the time varying unknown depth of the object from the centre of projection. Absolute translational and structural parameters, which are required in most robotics applications, can only be recovered if accurate depth information is available through an alternative source. However, if depth information is available, then because this data would most likely be contaminated with noise, Broida and Chellappa's technique as reviewed in Chapter 4 would not be a feasible approach. Object depth states and corresponding stochastic properties, if available, can be included in motion and measurement models, and normalization of other translational and structural parameters can be removed. This practice would simplify the dynamic model and allow structural parameters to be modeled by constants which could be decoupled from the state vector during occlusion or upon reaching some predefined threshold confidence level.

In the planar motion problem, feature points lie at constant heights above the plane of motion. A fundamental assumption of the proposed technique for planar motion with unknown structure is that the height of one feature point from the plane of motion is known. This assumption is really only required in initial tracking of the object and estimation of its structure. Once an estimate of the height of any feature point, perhaps one that does not appear to be subject to occlusion, reaches a predefined confidence level, the height of this feature point can be decoupled from the state vector and is thereafter assumed to be known.

Evaluation of motion and structure estimation schemes on simulated imagery provides an efficient means to verify correct behavior of filtering algorithms. Simulation studies of the proposed technique for planar motion have shown promising results in these initial investigations. Further analysis with simulated imagery is required to extend dynamical models to higher-order motion, develop techniques to allow filter decoupling during feature point occlusion and once structure estimates reach some predefined confidence level, and to investigate filter performance in the presence of egomotion. Extensions to higher-order, planar-constrained motion can retain a linear dynamic model and, particularly in the case of unknown structure, it may be possible to select one feature point as the origin of the object-centred refer-

ence frame. Adaptive construction of the structure state vector and the covariance matrix may lead to improved performance in the presence of feature occlusion over long time periods. This practice would also allow convergent structure estimates to be removed from the state vector in order that structure associated with new feature points could be estimated without incurring substantial computational loads by considering all possible feature points of interest in a single state vector. With this approach extensive knowledge of object structure could be acquired incrementally, provided active control of the imaging sensor is employed to maintain the object of interest in the field of view over a long time period.

Object motion and structure estimation in the presence of egomotion represents an interesting extension of the proposed technique in that sensor motion in fact becomes another state estimation problem. Simulation studies of the egomotion problem might, at least initially, assume that camera motion is known precisely in order to investigate the effects of sensor motion on object motion and structure estimation. It should be noted that both camera and object motion will most likely increase blur in the imagery and hence higher feature position measurement noise levels should be expected. Further investigations should consider the degree of coupling between camera motion state estimation errors and that associated with object motion and structure states (cross-covariance terms in an overall covariance matrix). Such coupling may result from time propagation of the covariance matrix even if camera and object motion are independent. Repeated measurement updates of the covariance matrix may also lead to coupling. Results of this investigation would determine the feasibility of operating separate filters for camera and object state estimation. In any case, state feedback of camera and object motion estimation to camera control actuators in order to maintain track on the object represents an interesting and challenging problem in optimal control theory. Navigation for autonomous vehicles through image-based tracking of known environmental features represents yet another direction for future research which could be formulated as an extension of the proposed technique.

Application of the proposed technique to real imagery will be a primary focus of future work. In such applications, the motion and structure estimation component,

which has shown promising results in simulation studies presented here, must be provided with feature position measurements and correspondence information. In static image processing, feature detection algorithms commonly operate in a global manner on the entire image. In dynamic scene interpretation, however, the location of important features can be predicted from the interpretation of previous images. This is particularly true of the Kalman filtering approach which provides natural prediction capabilities. The expected position of a feature in the next image sample as well as an estimate of state prediction error allows a small region to be defined in this future image within which the feature of interest will be found with high probability. Since nearly all relevant information in the image is contained in a limited number of small regions, concentration of all computing power on these regions could lead to very efficient solutions to the feature detection and correspondence problems. Controlled correlation techniques proposed by Dickmanns and Graefe [37] may prove to be a suitable approach to this problem.

Developments of Chapter 4 formulated motion models and investigated filter performance for constant translational and rotational velocity. It has been noted that neither technique was generally applicable to the case of pure translational motion. If it is known *a priori* that motion is translational only, estimation of angular velocity can be removed, and one feature point can be selected as the origin of the object-centred frame and hence has known structure. In general, these methods can be applied only if an initialization scheme is available which will detect the presence or absence of rotational motion. However, good performance can only be expected if angular velocity is exactly zero, or is relatively high; object trajectories with "small" angular rates will most likely lead to serious degradation in position and structure estimation. It may be possible, however, to construct an alternative motion and structural model appropriate to the case of small angular rates. This suggests that applications of this technique to general motion analysis problems, at least during an initialization stage, may benefit from an adaptive filtering approach similar in form to that proposed by Chang *et. al.* [33].

Future success in applications of image-based motion analysis techniques to machine vision will be largely dependent on the ability to exploit all available prior

knowledge about the problem of interest. This thesis has provided initial investigations of a technique to incorporate knowledge of planar motion into the state estimation problem. The Kalman filter formulation is based on a measurement model and dynamic constraints which are described through a system of differential equations. In many applications, however, additional prior knowledge about the observed process may also be expressed in the form of mathematical constraints. For example, it may be known *a priori* that the true state $\mathbf{x}$ must lie on a hypersurface $C(\mathbf{x}) = 0$ in state-space. Another class of prior information may be expressed in the form of interval constraints $\mathbf{x} \in \mathcal{D}$ where the set $\mathcal{D}$ in state-space is specified in the formulation of the problem. A planar motion constraint can be expressed in the form $C(\mathbf{x}) = 0$, but in the proposed technique this constraint has been incorporated directly into the motion model. Motion on a non-planar surface might also be expressed in the form $C(\mathbf{x}) = 0$, but direct substitution into the motion model may be neither simple nor well-behaved due to non-linearities. Interval constraints, on the other hand, may arise from prior knowledge concerning the expected size of the object, or its expected position in the environment. Extension of recursive state estimation to include these alternate forms of constraints represents a challenging area for future research.

References

- [1] A. Rosenfeld, "Survey - Image analysis and computer vision: 1989," *Computer Vision, Graphics, and Image Processing*, Vol. 50, 1990, pp 188-240.
- [2] W.N. Martin, J.K. Aggarwal, "Survey: Dynamic scene analysis," *Computer Graphics and Image Processing*, Vol. 7, 1978, pp 356-374.
- [3] H. Nagel, "Analysis techniques for image sequences," in *Proceedings of the Fourth International Conference on Pattern Recognition*, Kyoto, Japan, Nov. 7-10, 1978, pp 186-211.
- [4] H. Nagel, "Image sequences - Ten (octal) years - From phenomenology towards a theoretical foundation," in *Proceedings of the Eight International Conference on Pattern Recognition*, Paris, France, Oct. 27-31, 1986, pp 1174-1185.
- [5] J.K. Aggarwal, "Motion and time-varying imagery - An Overview," in *Proceedings, Workshop on Motion: Representation and Analysis*, Charleston, S.C., May 7-9, 1986, pp 1-6.
- [6] J.K. Aggarwal, L.S. Davis, W.N. Martin, "Correspondence processes in dynamic scene analysis," *Proceedings of the IEEE*, Vol. 69, No. 5, May 1981, pp 562-572.
- [7] J.K. Aggarwal, A. Mitiche, "Structure and motion from images: fact and fiction," in *Proceedings, Third Workshop on Computer Vision: Representation and Control*, Oct. 13-16, Bellaire, Michigan, 1985, pp 127-128.
- [8] B.G. Schunck, "The image flow constraint equation," *Computer Vision, Graphics, and Image Processing*, Vol. 35, 1986, pp 20-46.

- [9] B.G. Schunck, "Image flow segmentation and estimation by constraint line clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-11, No. 10, Oct. 1989, pp 1610-1627.
- [10] K. Prazdny, "On the information in optical flows," *Computer Vision, Graphics, and Image Processing*, Vol. 22, 1983, pp 239-259.
- [11] A.M. Waxman, S. Ullman, "Surface structure and three-dimensional motion from image flow kinematics," *International Journal of Robotics Research*, Vol. 4, No. 3, Fall 1985, pp 72-94.
- [12] K. Kanatani, "Structure and motion from optical flow under perspective projection," *Computer Vision, Graphics, and Image Processing*, Vol. 38, 1987, pp 122-146.
- [13] B.K.P. Horn, B.G. Schunck, "Determining optical flow," *Artificial Intelligence*, Vol. 17, 1981, pp 185-203.
- [14] 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. PAMI-7, No. 4, July 1985, pp 384-401.
- [15] J.W. Roach, J.K. Aggarwall, "Determining the movement of objects from a sequence of images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-2, No. 6, Nov. 1980, pp 554-562.
- [16] R.Y. Tsai, T.S. Huang, "Estimating three-dimensional motion parameters of a rigid planar patch III: Finite point correspondences and the three view problem," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP- 2, No. 2, Apr. 1984, pp 213-220.
- [17] R.Y. Tsai, T.S. Huang, "Uniqueness and estimation of three dimensional motion parameters of rigid objects with curved surfaces," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-6, No. 1, Jan. 1984, pp 13-26.

- [18] J. Fang, T.S. Huang, "Some experiments on estimating the 3-D motion parameters of a rigid body from two consecutive image frames," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-6, No. 5, Sept. 1984, pp 545-554.
- [19] R.P. Futrelle, G.C. Speckert, "Extraction of motion data by interactive image processing," in *Proceedings, IEEE Conference on Pattern Recognition and Image Processing*, Chicago, Illinois, May 31-June 2, 1978, pp 405-408.
- [20] D.B. Gennery, "Tracking known three-dimensional objects," in *Proceedings, National Conference on Artificial Intelligence, AAAI-82*, Pittsburgh, Pennsylvania, Aug. 18-20, 1982, pp 13-17.
- [21] J. Hallam, "Resolving observer motion by object tracking," in *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, West Germany, Aug. 8-12, 1983, pp 792-797.
- [22] G.R. Legters, T.Y. Young, "A mathematical model for computer image tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-4, No. 6, Nov. 1982, pp 583-594.
- [23] J. Stuller, G. Krishnamurthy, "Kalman filter formulation of low-level television image motion estimation," *Computer Vision, Graphics, and Image Processing*, Vol. 21, 1983, pp 169-204.
- [24] T. Broida, R. Chellappa, "Estimation of object motion parameters from noisy images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-8, No. 1, Jan. 1986, pp 90-99.
- [25] T. Broida, R. Chellappa, "Kinematics and structure of a rigid object from a sequence of noisy images," in *Proceedings, Workshop on Motion: Representation and Analysis*, Charleston S.C., May 7-9, 1986, pp 95-100.
- [26] T. Broida, R. Chellappa, "Kinematics of a rigid object from a sequence of noisy images: a batch approach," in *Proceedings, Computer Vision and Pattern Recognition*, Miami Beach, Fla., June 22-26, 1986, pp 176-182.

- [27] T. Broida, R. Chellappa, "Performance bounds for estimating three dimensional motion parameters from a sequence of noisy images," *Journal of the Optical Society of America*, Vol. 6, No. 6, June 1989, pp 879-889.
- [28] T. Broida, S. Chandrashekar, R. Chellappa, "Recursive 3-D motion estimation from a monocular image sequence," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 26, No. 4, July 1990, pp 639-656.
- [29] G.J. Young, R. Chellappa, "3-D motion estimation using a sequence of noisy stereo images," in *Proceedings, Computer Vision and Pattern Recognition*, Ann Arbor, Michigan, June 5-9, 1988, pp 710-716.
- [30] G.J. Young, R. Chellappa, "3-D motion estimation using a sequence of noisy stereo images: models, estimation, and uniqueness results," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-12, No. 8, Aug. 1990, pp 735-759.
- [31] T.J. Broida, "Estimating the kinematics and structure of a moving object from a sequence of images," Ph.D. dissertation, University of Southern California, Los Angeles, 1987.
- [32] J.J. Wu, R.E. Rink, T.M. Caelli, V.G. Gourishankar, "Recovery of the 3-D location and motion of a rigid object through camera image (an extended Kalman filter approach)," *International Journal of Computer Vision*, Vol. 3, 1988, pp 373-394.
- [33] K. Chang, H. Lee, C. Chung, "Adaptive tracking system for a manoeuvring target using images with correlated noises," *International Journal of Systems Science*, Vol. 20, No. 5, 1989, pp 839-857.
- [34] N. Ayache, O. Faugeras, "Maintaining representations of the environment of a mobile robot," *IEEE Transactions on Robotics and Automation*, Vol. 5, No. 6, Dec. 1989, pp 804-819.

- [35] L. Matthies, T. Kanade, "The cycle of uncertainty and constraint in robot perception," in *Proceedings of the Fourth International Symposium on Robotics Research*, 1987, pp 327-335.
- [36] L. Matties, T. Kanade, R. Szeliski, "Kalman filter-based algorithms for estimating depth from image sequences," *International Journal of Computer Vision*, Vol. 3, 1989, pp 209-236.
- [37] E.D. Dickmanns, V. Graefe, "Dynamic monocular machine vision," *International Journal of Machine Vision and Applications*, Springer-Verlag, New York, Vol. 1, No. 4, 1988, pp 223-240.
- [38] E.D. Dickmanns, V. Graefe, "Applications of dynamic monocular machine vision," *International Journal of Machine Vision and Applications*, Springer-Verlag, New York, Vol. 1, No. 4, 1988, pp 241-261.
- [39] V. Graefe, "A pre-processor for the real-time interpretation of dynamic scenes," in T.S. Huang (Ed.), *Image Sequence Processing and Dynamic Scene Analysis*, Springer-Verlag, Berlin, 1983, pp 519-531.
- [40] H.G. Meissner, E.D. Dickmanns, "Control of an unstable plant by computer vision," in T.S. Huang (Ed.), *Image Sequence Processing and Dynamic Scene Analysis*, Springer-Verlag, Berlin, 1983, pp 532-548.
- [41] R. Schell, E.D. Dickmanns, "Autonomous automatic landing through computer vision," in *AGARD Conference Proceedings No. 455*, Guidance and Control Panel 48th Symposium, Instituto da Defesa Nacional in Lisbon, Portugal, May 9-12, 1989, pp 24:1-24:9.
- [42] E.D. Dickmanns, B. Mysliwitz, T. Christians, "An integrated spatio-temporal approach to automatic guidance of autonomous vehicles," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 20, No. 6, Nov./Dec. 1990, pp 1273-1284.

- [43] R.E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions ASME, Ser. D: Journal of Basic Engineering*, Vol. 82, 1960, pp 35-45.
- [44] J.H. Taylor, "The Cramer-Rao estimation error lower bound computation for deterministic nonlinear systems," *IEEE Transactions on Automatic Control*, Vol. AC-24, No.2, April 1979, pp 313-314.
- [45] C. Chang, J. Tabaczynski, "Application of state estimation to target tracking," *IEEE Transactions on Automatic Control*, Vol. AC-29, No. 2, Feb. 1984, pp 98-109.
- [46] M.H. Ang, V.D. Tourassis, "Singularities of Euler and roll-pitch-yaw representations," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. AES-23, No. 3, May 1987, pp 317-324.
- [47] I.Y. Bar-Itzhack, Y. Oshman, "Attitude determination from vector observations: quaternion estimation," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. AES-21, No. 1, Jan. 1985, pp 128-135.
- [48] J. Stuelpnagel, "On the parameterization of the three-dimensional rotation group," *SIAM Review*, Vol. 6, No. 4, Oct. 1964, pp 422-430.
- [49] Y.L. Gu, "An exploration of orientation representation by Lie algebra for robotic applications," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 20, No. 1, Jan. 1990, pp 243-248.
- [50] S. Ullman, *The Interpretation of Visual Motion*, Cambridge, MA: M.I.T. Press, 1979.
- [51] A.H. Jazwinski, *Stochastic Processes and Filtering Theory*, New York: Academic, 1970.
- [52] A. Gelb, *Applied Optimal Estimation*, Cambridge MA: M.I.T. Press, 1974.
- [53] B.D.O. Anderson, J.B. Moore, *Optimal Filtering*, Englewood Cliffs, NJ: Prentice-Hall, 1979.

- [54] P.S. Maybeck, *Stochastic Models, Estimation, and Control*, New York: Academic, Vol. 1, 1979, and Vol. 2, 1982.
- [55] A.V. Balakrishnan, *Kalman Filtering Theory*, Optimization Software Inc., Publications Division, New York, 1984.
- [56] C.R. Rao, *Linear Statistical Inference and Its Applications, Second Edition*, New York: John Wiley and Sons, 1973.
- [57] K.V. Bury, *Statistical Models in Applied Science*, New York: John Wiley and Sons, 1975.
- [58] N.U. Ahmed, *Elements of Finite-Dimensional Systems and Control Theory*, Pittman Monographs and Surveys in Pure and Applied Mathematics, 37, Longman Scientific and Technical: Harlow Essex, England, 1988.
- [59] B. Etkin, *Dynamics of Atmospheric Flight*, New York: John Wiley and Sons, 1972.

Appendix A

Simulation Software

All “proof-of-concept” simulation software employed in this study, with the exception of imagery generation software, was written for the MatLab¹ CAD package. MatLab provides immediate access to powerful matrix analysis routines through relatively simple syntax which, in fact, can be read as pseudo-code algorithms for eventual implementation in executable programs. With these tools, a design engineer can focus on the design process for the problem at hand rather than on tedious programming of matrix manipulation routines which have been programmed many times before. Of course, such tedious programming is eventually required for final implementation, however, generation of efficient code is often highly application specific.

Over 40 MatLab script files (programs) were employed during these simulation studies. Most of these files were designed to parse data storage arrays and display intermediate and final results. Corresponding to each script file with filtering code, an associated script file was required to compute Cramer-Rao bounds. For brevity, only a small subset of these files will be listed here. The particular camera viewing angles investigated in Chapter 3 are a special case of the more general estimation problems considered in Chapter 4. Moreover, code associated with the known structure case of the proposed technique for planar motion in Chapter 4 is essentially included in that for both motion and structure estimation. Computation of Cramer-Rao bounds follows very closely the same algorithm as that associated with

¹Copyright, The MathWorks, Inc.

the Kalman covariance matrix, except that Fisher's information matrix is computed from the true object trajectory as shown in Section 2.4.

All simulation analysis of this thesis employed computer-generated imagery. MatLab script files associated with Broida and Chellappa's technique include imagery generation without occlusion for a translating and rotating block. Corresponding imagery with occlusion employed in analysis of the proposed technique for planar motion was generated by a program written for the Turbo² C programming language. This program is currently configured for constant translational and rotational velocities in a known plane with a stationary camera, however, it could easily be extended to higher-order object motion and a moving sensor. Standard function calls to the Borland Graphics Interface (BGI) library are employed to display the image sequence. Output from this program includes image feature positions with correspondence and occlusion information.

This appendix contains the following code listings:

Broida and Chellappa's Technique - This implementation is entirely in MatLab and includes BC90.M, the parent script file, and three functions: time propagation of the true state with measurement computations, TSTEPBC.M; computation of the system matrix $f(x)$ and its Jacobian $F(x)$, GET_FBC.M; and computation of the measurement matrix $h(x)$ and its Jacobian $H(x)$, GET_HBC.M.

Imagery Generation - The Turbo C program for imagery generation with or without occlusion, BL_WORLD.C, is menu-driven for ease of use. This program will run on any personal computer with at least two-page graphics capabilities (Hercules, EGA, VGA) and requires Turbo C version 2.0 or higher with the corresponding BGI driver.

Proposed Planar Motion Technique - This implementation loads imagery information from files produced with BL_WORLD.C and executes a Monte Carlo simulation of the proposed technique. Only code for both motion and structure

²Trademark of Borland International.

estimation with local iterations is demonstrated here.

All listings include extensive comments which describe program variables, data flow, and general algorithms employed.

A.1 Broida and Chellappa's Technique

This listing is the parent script file for Broida and Chellappa's method of [28].

```
%BC90.M - MatLab script file to simulate Broida and Chellappa's 1990
%          method. This implementation includes initialization for
%          planar motion.

clear;                                %clear all workspace variables
rand('normal');                        %Gaussian random deviate generation
max_MC = 1;                            %No. runs in Monte Carlo simulation
num_alt = 0;                           %Number of non-convergent runs
max_alt = 20;                          %Max non-convergent runs allowed

num_match = 4;                         %Number of feature points
max_it = 10;                           %Max iterations per measurement event
threshold = 0.01;                      %Threshold to terminate iterations

start = 0;                             %Initial time
stop = 20;                             %Stop time
T = 1/10;                              %Sample period
t = start:T:stop;                      %Time vector
[garb,numsteps] = size(t);             %Get number of steps

sig_z = 0.02;                          %Define measurement noise levels
R = sig_z*sig_z*eye(2*num_match);      %Measurement noise covariance

%Define system parameters

                                %Feature points in Fo
r_match = [ 5.0,  5.0, -5.0, -5.0, -5.0, -5.0,  5.0,  5.0;
            10.0, 10.0,  10.0, 10.0, -10.0, -10.0, -10.0, -10.0;
            0.0,-10.0, -10.0,  0.0,  0.0, -10.0, -10.0,  0.0];
%          p1  p2   p3   p4   p5   p6   p7   p8

%***** Initialization for Planar Motion *****

TEC_E = [0;-30;-30];                  %Camera position in Fe
C_ang = pi*[45;-45]/180.0;            %Camera pan (yaw) and tilt (pitch)

c_temp = cos(C_ang(2));               %Compute IEC
s_temp = sin(C_ang(2));
G2_tilt = [c_temp, 0, -s_temp;        %Tilt or pitch rotation
           0, 1, 0;
           s_temp, 0, c_temp];
c_temp = cos(C_ang(1));
s_temp = sin(C_ang(1));
```

```

G3_pan = [c_temp,s_temp,0;           %Pan or yaw rotation
          -s_temp,c_temp,0;
          0,0,1];
IEC = G2_tilt*G3_pan;                %Rotation from Fe to Fc

TEC_C = IEC*TEC_E;                   %Translation from Fe to Fc in Fc

x_E = [30;-25;0];                   %Initial posn of origin of Fo wrt Fe
v_E = [2.5;5;0];                    %Trans velocity of Fo wrt Fe
psi_E = 1;                          %Initial orientation of Fo wrt Fe
w_E = [0;0;-0.5];                   %Angular velocity of Fo wrt Fe

                                     %Compute IC's wrt Fc and quaternions
x_C = IEC*x_E - TEC_C;              %Initial posn of origin of Fo wrt Fc
v_C = IEC*v_E;                      %Trans velocity of Fo wrt Fc
w_C = IEC*w_E;                      %Angular velocity of Fo wrt Fc
IOE_0 = [cos(psi_E),-sin(psi_E),0; %Initial rotation matrix Fo -> Fe
         sin(psi_E), cos(psi_E),0;
         0,0,1];
IOC_0 = IEC*IOE_0                   %Rotation matrix Fo -> Fc
gam_0 = acos((trace(IOC_0)-1)/2); %Angle relating Fo -> Fc
xi_sharp_0 = (IOC_0 - IOC_0')/(2*sin(gam_0)); %Cross product operator
xi_0 = [-xi_sharp_0(2,3);xi_sharp_0(1,3);-xi_sharp_0(1,2)];
xi_0 = xi_0/norm(xi_0);             %Initial axis of rotation
s_gam_2 = sin(gam_0/2);
q_0 = [-xi_0(1)*s_gam_2; -xi_0(2)*s_gam_2;..
        -xi_0(3)*s_gam_2;cos(gam_0/2)]; %Initial quaternion of rotation

IC_state = [x_C(2:3);v_C;q_0;w_C]; %Initial motion state vector
IC_depth = x_C(1);
%***** End Initialization for Planar Motion *****

%True unnormalized initial full state vector. The true state is
%propagated in time in unnormalized form.

x0_true = [IC_state;r_match(:,3);r_match(:,7);r_match(:,1);r_match(:,5)];
depth = IC_depth; %true depth(0)

%Assumed covariance for normalized initial state
std_x0 = [0.1; 0.1;                %y/x and z/x states
          0.05;0.05;0.05;          %Velocity/depth states
          0.0; 0.0; 0.0; 0.0;      %Quaternions
          0.2; 0.2; 0.2;           %Angular velocity
          (5/depth)*ones(3*num_match,1)]; %Structure
cov = diag(diag(std_x0*std_x0')); %Initial covariance matrix

```

```

%Define maximum error to check for apparent divergence
max_error = [5; 5; %y/x and z/x states
             5; 5; 5; %Velocity/depth states
             1.5;1.5;1.5;1.5; %Quaternions
             0.4;0.4;0.4; %Angular velocity
             10*ones(3*num_match,1)]; %Structure

%***** Begin Monte Carlo Simulation *****
j = 1; %Initialize loop control
while (j <= max_MC), %Main loop of MC simulation
    filt_alt_flag = 0; %Reset apparent divergence flag
    %Initialize filter state
    if (max_MC == 1), %if not doing Monte-Carlo run
        xk_0 = [IC_state(1)-3;IC_state(2)+3;0;0;0;q_0;0;0;0;...
                zeros(3*num_match,1)];
        xk_0(1:5) = xk_0(1:5)/depth; %Normalize by depth
        xk_0(13:3*num_match+12) = xk_0(13:3*num_match+12)/depth;
    else
        x0_tmp = x0_true; %Normalize true initial state
        x0_tmp(1:5) = x0_true(1:5)/depth;
        x0_tmp(13:3*num_match+12) = x0_true(13:3*num_match+12)/depth;
        %IC's = sample from N(x0_true,cov)
        xk_0 = x0_tmp + std_x0 .* rand(x0_tmp);
        xk_0(3*num_match+12) = 0; %Last state known exactly as 0
    end;

    x_true = x0_true; %Initialize true state
    xk_full = xk_0; %Initialize filter state

    xk = xk_full(1:3*num_match+11); %Remove last state (known) from xk
    cov = cov(1:3*num_match+11,1:3*num_match+11); % and cov

    %Begin simulation
    %Take initial measurements
    [x_true,z,depth,Ioc_true] = tstepbc(x_true,num_match,0,depth);
    z = z + sig_z*rand(z); %Add noise

    x_old = xk; %Prepare for iteration
    done = 0; %Initially false
    i = 1;
    while ((~done) & (i <= max_it)),
        x_prev = xk; %Save previous iterate
        xk_full(1:3*num_match+11) = xk; %Compute innovation nu, and H
        [H,h] = get_hbc(xk_full,num_match);
        nu = z-h;
        H = H(:,1:3*num_match+11); %Remove last column of H
    end
end

```

```

Kk = cov*H'*inv(H*cov*H' + R);           %Compute Kalman gain
xk = x_old + Kk*(nu - H*(x_old - xk)); %State update
i=i+1;
if (max(abs(x_prev-xk)) < threshold), %Check for termination
    done = 1;
end; %If
end; %While
cov = (eye(3*num_match+11) - Kk*H)*cov; %update covariance

q = xk(6:9);                             %Normalize quaternion estimate
q = q/norm(q);
xk(6:9) = q;

%compute Ioc_hat from normalized q
Ioc_hat = [q(1)*q(1)-q(2)*q(2)-q(3)*q(3)+q(4)*q(4),...
            2*(q(1)*q(2)+q(3)*q(4)),...
            2*(q(1)*q(3)-q(2)*q(4)),...
            2*(q(1)*q(2)-q(3)*q(4)),...
            -q(1)*q(1)+q(2)*q(2)-q(3)*q(3)+q(4)*q(4),...
            2*(q(2)*q(3)+q(1)*q(4)),...
            2*(q(1)*q(3)+q(2)*q(4)),...
            2*(q(2)*q(3)-q(1)*q(4)),...
            -q(1)*q(1)-q(2)*q(2)+q(3)*q(3)+q(4)*q(4)];

xk_full(1:3*num_match+11) = xk; %Restore xk_full

tmp = x_true;                             %Store errors
tmp(1:5) = tmp(1:5)/depth;
tmp(13:3*num_match+12) = tmp(13:3*num_match+12)/depth;
error_hist = xk_full - tmp;
depth_hist = depth;

%Compute q convergence performance measure
J = sqrt(trace((Ioc_hat-Ioc_true)'*(Ioc_hat-Ioc_true)));
J_hist = 0;

for k = 2:numsteps, %Start simulation loop
    disp([k,j,num_alt]); %Feedback to user
    %Step true state and get measurements
    [x_true,z,depth,Ioc_true] = tstepbc(x_true,num_match,T,depth);
    z = z + sig_z*rand(z); %Add noise

    [F,f] = get_fbc(xk_full,num_match); %Compute prediction
    F = F(1:3*num_match+11,1:3*num_match+11);
    f = f(1:3*num_match+11);
    xk = xk_full(1:3*num_match+11); %Remove last state

```

```

phi = expm(F*T);           %Approx state transition matrix
xk = xk + T*f;             %Euler integration of system
cov = phi*cov*phi';        %Compute prediction covariance

x_old = xk;                %Prepare for iteration
done = 0;                  %False
i = 1;
while ((~done) & (i <= max_it)),
    x_prev = xk;           %Store previous iterate
                           %Compute innovation nu, and H
    xk_full(1:3*num_match+11) = xk;
    [H,h] = get_hbc(xk_full,num_match);
    nu = z-h;
    H = H(:,1:3*num_match+11); %Remove last column of H
    Kk = cov*H'*inv(H*cov*H' + R); %Compute Kalman gain
    xk = x_old + Kk*(nu - H*(x_old - xk)); %State update
    i=i+1;
    if (max(abs(x_prev-xk)) < threshold),
        done = 1;
    end; %If
end; %While
cov = (eye(3*num_match+11) - Kk*H)*cov; %update covariance

q = xk(6:9);               %Normalize quaternion estimate
q = q/norm(q);
xk(6:9) = q;
xk_full(1:3*num_match+11) = xk; %restore xk_full
                           %Compute Ioc_hat from normalized q
Ioc_hat = [q(1)*q(1)-q(2)*q(2)-q(3)*q(3)+q(4)*q(4),...
            2*(q(1)*q(2)+q(3)*q(4)),...
            2*(q(1)*q(3)-q(2)*q(4)),...
            2*(q(1)*q(2)-q(3)*q(4)),...
            -q(1)*q(1)+q(2)*q(2)-q(3)*q(3)+q(4)*q(4),...
            2*(q(2)*q(3)+q(1)*q(4)),...
            2*(q(1)*q(3)+q(2)*q(4)),...
            2*(q(2)*q(3)-q(1)*q(4)),...
            -q(1)*q(1)-q(2)*q(2)+q(3)*q(3)+q(4)*q(4)];

tmp = x_true;              %Store errors
tmp(1:5) = tmp(1:5)/depth;
tmp(13:3*num_match+12) = tmp(13:3*num_match+12)/depth;
error = xk_full - tmp;
depth_hist = [depth_hist,depth];
error_hist = [error_hist,error];

%Compute q convergence performance measure

```

```

J = sqrt(trace((Ioc_hat-Ioc_true)'*(Ioc_hat-Ioc_true)));
J_hist = [J_hist,J];

error = abs(error);           %Check for apparent divergence
if ((max_MC > 1) & (k > 2*numsteps/3) & ..
    (max(error-max_error) > 0.0)),
    filt_alt_flag = 1;
    num_alt = num_alt + 1;
    j = j - 1;
    disp('Filter did not converge to true parameters.');
```

break;

```

end; %If
end; %For k

if (num_alt > max_alt),
    disp('Maximum number of non-convergent runs exceeded!');
```

break;

```

end; %If

if ((max_MC > 1) & (~filt_alt_flag)), %Update statistics of run
    if (j == 1),
        mean_hist = error_hist/max_MC;
        sum_sqr = (error_hist .* error_hist)/max_MC;
        mean_J = J_hist/max_MC;
        sqr_J = (J_hist .* J_hist)/max_MC;
    else
        mean_hist = mean_hist + error_hist/max_MC;
        mean_J = mean_J + J_hist/max_MC;
        sum_sqr = sum_sqr + (error_hist .* error_hist)/max_MC;
        sqr_J = sqr_J + (J_hist .* J_hist)/max_MC;
    end; %Else
end; %If

j = j+1;
end; %While j<= max_MC loop

%***** END OF BC90.M *****
```

```
function [xout,z,depth_out,Ioc] = tstepbc(x,Np,T,depth);
```

```

rout = x; %Initialize output vector
depth_out = depth + T*x(3); %Translational motion
rout(1) = x(1) + T*x(4);
rout(2) = x(2) + T*x(5);

```

```
if (mag_w > 1.0E-5), %Form Omega matrix
```

```
Omega = Omega/mag_w;
```

```
q = x(6:9); %Quaternion time update
```

```
tmp = mag_w*T/2.0;
```

```
q = (cos(tmp)*eye(4) + 2*sin(tmp)*Omega)*q;
```

```
xout(6:9) = q;
```

```
end; %If
```

```
%Now compute image points
```

```
S = zeros(3,Np);
```

```
%Get structure matrix S = [r1...rNp]
```

```
for i=1:Np,
```

```
S(:,i) = x(3*i+10:3*i+12);
```

end: %For

```
q = xout(6:9);
```

```
%Get quaternion q from xout
```

%Compute Ioc from q

$$I_{oc} = [q(1)*q(1)-q(2)*q(2)-q(3)*q(3)+q(4)*q(4), \dots]$$
$$2*(q(1)*q(2)+q(3)*q(4)), \dots$$

```
2*(q(1)*q(3)-q(2)*q(4));
```

$$2*(q(1)*q(2)-q(3)*q(4)), \dots$$
$$-q(1)*q(1)+q(2)*q(2)-q(3)*q(3)+q(4)*q(4), \dots$$

```
2*(q(2)*q(3)+q(1)*q(4));
```

$$2*(q(1)*q(3)+q(2)*q(4)), \dots$$
$$2*(q(2)*q(3)-q(1)*q(4)), \dots$$

```
-q(1)*q(1)-q(2)*q(2)+q(3)*q(3)+q(4)*q(4)];
```

$$Sc = Ioc * S;$$

%Compute structure, Sc, in Fc

```
z = zeros(2*Np,1);           %Compute image feature points z
for i=1:Np,
    z(2*i-1) = (xout(1) + Sc(2,i))/(depth_out + Sc(1,i));
    z(2*i)   = (xout(2) + Sc(3,i))/(depth_out + Sc(1,i));
end;

%***** END TSTEPBC.M *****
```


This listing describes computations of the system vector $f(x)$ and its Jacobian $F(x)$ employed in BC90.M for Broida and Chellappa's technique of [28].

```
function [F,f] = get_FBC(x,Np);

% Function to return dx/dt = f(x) and df/dx = F given the state x and
% the number of feature points Np for Broida and Chellappa's 1990
% method.

f = zeros(x);                                %Compute f(x)
f(1) = x(4)-x(1)*x(3);
f(2) = x(5)-x(2)*x(3);
f(3) = -x(3)*x(3);
f(4) = -x(4)*x(3);
f(5) = -x(5)*x(3);
f(6) = 0.5*(-x(12)*x(7)+x(11)*x(8)-x(10)*x(9));
f(7) = 0.5*(x(12)*x(6)-x(10)*x(8)-x(11)*x(9));
f(8) = 0.5*(-x(11)*x(6)+x(10)*x(7)-x(12)*x(9));
f(9) = 0.5*(x(10)*x(6)+x(11)*x(7)+x(12)*x(8));
f(13:3*Np+12) = -x(3)*x(13:3*Np+12);

F1=-x(3)*eye(5);                            %Compute F(x) by formation of
F1(:,3) = -x(1:5);                          % four block components
F1(3,3)= 2*F1(3,3);
F1(1,4) = 1.0;
F1(2,5) = 1.0;

F2 = zeros(7,7);
F2(1,2) = -x(12);
F2(1,3) = x(11);
F2(1,4) = -x(10);
F2(1,5) = -x(9);
F2(1,6) = x(8);
F2(1,7) = -x(7);

F2(2,1) = x(12);
F2(2,3) = -x(10);
F2(2,4) = -x(11);
F2(2,5) = -x(8);
F2(2,6) = -x(9);
F2(2,7) = x(6);

F2(3,1) = -x(11);
F2(3,2) = x(10);
F2(3,4) = -x(12);
F2(3,5) = x(7);
```

```
F2(3,6) = -x(6);
F2(3,7) = -x(9);

F2(4,1) = x(10);
F2(4,2) = x(11);
F2(4,3) = x(12);
F2(4,5) = x(6);
F2(4,6) = x(7);
F2(4,7) = x(8);

F2 = 0.5*F2;

F3 = zeros(3*Np,5);
F3(:,3) = -x(13:3*Np+12);

F4 = -x(3)*eye(3*Np);

F = [F1,zeros(5,7+3*Np);           %Build F(x)
     zeros(7,5),F2,zeros(7,3*Np);
     F3,zeros(3*Np,7),F4];

%***** END GET_FBC.M *****
```

This listing describes computations of the measurement vector $h(x)$ and its Jacobian $H(x)$ employed in BC90.M for Broida and Chellappa's technique of [28].

```
function [H,h,Ioc] = get_HBC(x,Np);
%Function to return h(x) and H = dh(x)/dx for Broida and Chellappa's
%1990 method as well as the estimated rotation matrix Ioc from Fo
%to Fc. Np = number of feature points.

S = zeros(3,Np);           %Get structure matrix S = [r1...rNp]
for i=1:Np,
    S(:,i) = x(3*i+10:3*i+12);
end; %For
q = x(6:9);               %Get quaternion q from x
                           %Compute Ioc from q
Ioc = [q(1)*q(1)-q(2)*q(2)-q(3)*q(3)+q(4)*q(4),...
        2*(q(1)*q(2)+q(3)*q(4)),...
        2*(q(1)*q(3)-q(2)*q(4));
        2*(q(1)*q(2)-q(3)*q(4)),...
        -q(1)*q(1)+q(2)*q(2)-q(3)*q(3)+q(4)*q(4),...
        2*(q(2)*q(3)+q(1)*q(4));
        2*(q(1)*q(3)+q(2)*q(4)),...
        2*(q(2)*q(3)-q(1)*q(4)),...
        -q(1)*q(1)-q(2)*q(2)+q(3)*q(3)+q(4)*q(4)];
Sc = Ioc*S;               %Compute structure, Sc, in Fc

h = zeros(2*Np,1);        %Compute h(x)
for i=1:Np,
    h(2*i-1) = (x(1) + Sc(2,i))/(1 + Sc(1,i));
    h(2*i)   = (x(2) + Sc(3,i))/(1 + Sc(1,i));
end; %For

dIdq1 = 2*[ q(1), q(2), q(3);
             q(2),-q(1), q(4);
             q(3),-q(4),-q(1)]; %Compute dIoc/dq matrices
dIdq2 = 2*[-q(2), q(1),-q(4);
             q(1), q(2), q(3);
             q(4), q(3),-q(2)];
dIdq3 = 2*[-q(3), q(4), q(1);
             -q(4),-q(3), q(2);
             q(1), q(2), q(3)];
dIdq4 = 2*[ q(4), q(3),-q(2);
             -q(3), q(4), q(1);
             q(2),-q(1), q(4)];

dScdq1 = dIdq1*S;         %Compute dSc/dq matrices
dScdq2 = dIdq2*S;
```

```

dScdq3 = dIdq3*S;
dScdq4 = dIdq4*S;

P = zeros(2*Np,2);           %Compute P = 1st two columns of H
for i=1:Np,
    P(2*i-1,1) = 1.0/(1 + Sc(1,i));
    P(2*i,2)   = 1.0/(1 + Sc(1,i));
end;

W = zeros(2*Np,4);           %Compute W = 6th to 10th columns of H,
                                %   W = dh/dq
    tmp1 = 1+Sc(1,i);
    sqr_tmp1 = tmp1*tmp1;
    tmp2 = x(1)+Sc(2,i);
    W(2*i-1,1) = (tmp1*dScdq1(2,i) - tmp2*dScdq1(1,i))/sqr_tmp1;
    W(2*i-1,2) = (tmp1*dScdq2(2,i) - tmp2*dScdq2(1,i))/sqr_tmp1;
    W(2*i-1,3) = (tmp1*dScdq3(2,i) - tmp2*dScdq3(1,i))/sqr_tmp1;
    W(2*i-1,4) = (tmp1*dScdq4(2,i) - tmp2*dScdq4(1,i))/sqr_tmp1;
    tmp2 = x(2)+Sc(3,i);
    W(2*i,1) = (tmp1*dScdq1(3,i) - tmp2*dScdq1(1,i))/sqr_tmp1;
    W(2*i,2) = (tmp1*dScdq2(3,i) - tmp2*dScdq2(1,i))/sqr_tmp1;
    W(2*i,3) = (tmp1*dScdq3(3,i) - tmp2*dScdq3(1,i))/sqr_tmp1;
    W(2*i,4) = (tmp1*dScdq4(3,i) - tmp2*dScdq4(1,i))/sqr_tmp1;
end; %For

B = zeros(2*Np,3*Np);        %Compute B = "dh/dS" = 13th to
                                %   3Np+12th columns of H
    tmp1 = 1+Sc(1,i);
    sqr_tmp1 = tmp1*tmp1;
    tmp2 = x(1)+Sc(2,i);
    B(2*i-1,3*i-2) = (tmp1*Ioc(2,1)-tmp2*Ioc(1,1))/sqr_tmp1;
    B(2*i-1,3*i-1) = (tmp1*Ioc(2,2)-tmp2*Ioc(1,2))/sqr_tmp1;
    B(2*i-1,3*i)   = (tmp1*Ioc(2,3)-tmp2*Ioc(1,3))/sqr_tmp1;
    tmp2 = x(2)+Sc(3,i);
    B(2*i,3*i-2) = (tmp1*Ioc(3,1)-tmp2*Ioc(1,1))/sqr_tmp1;
    B(2*i,3*i-1) = (tmp1*Ioc(3,2)-tmp2*Ioc(1,2))/sqr_tmp1;
    B(2*i,3*i)   = (tmp1*Ioc(3,3)-tmp2*Ioc(1,3))/sqr_tmp1;
end;
H = [P,zeros(2*Np,3),W,zeros(2*Np,3),B]; %Build H

%***** END GET_HBC.M *****

```

A.2 Imagery Generation

The program BL_WORLD.C is driven by a system of menus. An opening demonstration screen is shown first. Hitting any key displays the main menu:

```

Image Sequence Generation: Planar Motion
  F1 - Camera Parameters
  F2 - Block Structure and Motion Parameters
  F3 - Output Enable (Currently OFF)
  F10 - Run Imagery Simulation
  ESC - QUIT

```

The F1 function key selection provides access to camera focal length, position, and orientation through the following submenu:

```

Current Camera Parameters:
F1 - Focal Length      = 0.003 m
F2 - Position (x, y, z) = (0, 0, -30) m
F3 - Angles (pan, tilt) = (0, -45) Deg.
ESC - Exit

```

Parameter to change: <your selection>

Both focal length and position are entered in meters (really a relative unit here) while pan (yaw) and tilt (pitch) angles are entered in degrees. The camera position is relative to the the earth-fixed frame F_E shown in Figure 4.17. Position or angle entries must be separated by whitespace characters (blanks, for example).

The F2 function key selection from the main menu provides access to block structure, initial position, motion, occlusion, and time sequence parameters through the sub-menu:

```

Current Block Parameters:
F1 - Dimension (depth x)   = 10 m
      (width y)           = 20 m
      (height z)          = 10 m
F2 - Initial Position (x,y) = (30, 0) m
F3 - Initial Yaw Angle      = 0 Deg.
F4 - Trans. Velocity (vx,vy) = (0, 0) m/s
F5 - Ang. Velocity (wz)     = -57.2958 Deg/s
F6 - Time (start,step,stop) = (0, 0.1, 10) Sec.
F7 - Toggle Occlusion (Currently ON)

```

ESC - Exit

Parameter to change: <your selection>

The dimension of the block (see Figure 4.4) is defined by its depth (x_O direction), width (y_O), and height (z_O). The origin of F_O is placed on the base of the block in the x - y plane of F_E . The initial position, orientation, translational velocity, and angular velocity are given with respect to F_E .

The F3 function key selection from the main menu allows the user to specify an output file for imagery data and enable the output process. In this way the user may view the image sequence and alter motion parameters before a physical write to disk. The F3 selection will prompt for a data filename (max 70 characters):

Output filename for measurements: <yourfilename>

If the designated file exists, the user is asked for confirmation prior to overwriting the file:

<yourfilename> already exists. Overwrite? <y,n,ESC>

Pressing "y" overwrites the existing file with new data, pressing "n" will cause the program to prompt for a new filename, and pressing ESC returns control to the main menu. The output file contains image feature locations with unity focal length, and correspondence and occlusion information. These files are read directly by the MatLab script file which simulates the proposed technique.

The F10 selection from the main menu produces the image sequence and writes data to the specified file if output is enabled. The initial image frame is displayed and the user has the option of aborting imagery generation (ESC) or proceeding with the run (any other key). If the block is not visible in this first frame, it may still be "visible" in the output data since displayed imagery is clipped by the image plane dimensions while output data is not.

```

/***** BL_WORLD.C *****/
/*
/*   This program was developed in support of simulation studies */
/* of motion and structure parameter estimation for objects present */
/* in monocular image sequences. These studies focussed on the case */
/* in which the object is constrained to lie on a known planar */
/* surface and was viewed from arbitrary angles above the plane of */
/* motion with and without feature point occlusion. This program */
/* generates monocular image sequences through central projection */
/* image formation with or without occlusion for a translating and */
/* rotating rectangular block which is constrained to lie on a */
/* plane. Constant translational and rotational velocity of the */
/* object, and a stationary camera are also assumed, however, only */
/* slight modifications are required to simulate higher-order motion*/
/* and imagery under egomotion. Feature points of interest are the */
/* eight vertices of the rectangular block. */
/*
/*   For simplicity, this program requires no input data from */
/* external files. A menu system is employed to give the user */
/* adequate access to camera, block structure, and block motion */
/* parameters of interest. A single output file is produced for each*/
/* simulation run (if output is enabled) and contains all relevant */
/* data for input to MatLab routines which simulate motion and */
/* structure estimation algorithms. MatLab files which use these */
/* output files read the ASCII data directly and parse this data */
/* into required matrices. */
/*
/*   Recompile of this program requires TURBO C Vsn 2.0 or */
/* higher with the LARGE compiler option. A PROJECT file must also */
/* be used in order to link BL_WORLD.OBJ with GRAPHICS.LIB and other*/
/* standard TURBO C libraries. This program was designed for the */
/* Hercules graphics adapter, but could be used with any graphics */
/* adapter which has at least two pages of graphics and is supported*/
/* by a Borland Graphics Interface (*.BGI files). Two graphics */
/* pages are used in "animation": erasure and drawing takes place */
/* on a hidden page while the other page is visible, then pages are */
/* switched. */
/*****

/***** INCLUDES *****/

#include <graphics.h>      /*Standard graphics header */
#include <io.h>             /*Extended IO header */
#include <math.h>           /*Standard math header */
#include <stdlib.h>         /*Standard library header */

```



```

block_type block;          /*Main block defining object          */
block_type block2;         /*Temporary block for demo page */
struct cam_parms           /*Camera parameters           */
{
    float F_len;           /*Focal length in meters (12.5 -> 75mm) */
    float IM_size[2];      /*Image plane size (horiz,vert) in meters*/
    float m_to_scr[2];     /*Meter to screen conversion (horiz,vert)*/
    float posn[3];         /*Coordinates of origin of Fc wrt Fe    */
    float C_ang[3];        /*Camera Euler angles (pan,tilt,roll)   */
    float Iec[3][3];       /*Rotation from Fe to Fc                */
}cam_parms;               /* Note: pan=yaw, tilt=pitch          */
int P_to_F[6][5] = {{0,1,2,3,0}, /*P_to_F[i][j]=feature point No.    */
                    {2,3,4,5,2}, /* of jth vertex of ith block        */
                    {4,5,6,7,4}, /* face, i.e. face 1 contains        */
                    {6,7,0,1,6}, /* feature points p0, p1, p2 & p3*/
                    {1,2,5,6,1}, /* etc.                               */
                    {3,4,7,0,3}};

/* NOTE: The thesis documentation and MatLab script files label
/*       feature points starting with p1 through p8. In C, vectors
/*       and matrices begin with index 0, so here feature points
/*       are labeled p0 through p7.

/***** FUNCTION PROTOTYPES *****/
void default_init(void);      /*Default block and camera init    */
void usr_interface(void);    /*Main menu system                  */
void initialize(void);       /*Graphics initialization           */
void get_cam_parms(void);    /*User input for camera & block    */
void get_block_parms(struct block_type *this_block); /*parameters */
void enable_output(void);    /*Enables data output/file write   */
void generate_imagery(void); /*Graphics control                  */
                             /*Initialize block parameters      */
void define_block(struct block_type *this_block,
                  int block_num,float *dimension_0,
                  float *posn_0,float *E_ang_0,
                  float *Tvel_0,float *Avel_0,
                  int occflag);
void transpose(float *R);     /*Matrix transposition              */
void get_rot(float *R,float *params); /*Computes rotation matrix        */
                             /*Draw block on hidden page        */
void draw_block(struct block_type *this_block,int page);
FILE *open_write_file(char *msg); /*Get output file & IO check      */
                             /*Move block on planar surface      */
void move_block(struct block_type *this_block,float sample_T);
                             /*Data output routine              */
void output_measurements(FILE *outfile,
                         struct block_type *this_block,int page);

```

```

/***** FUNCTION MAIN *****/

main()
{
    default_init();          /*Default camera and block init */
    usr_interface();         /*Involk menu system          */
}

/***** END FUNCTION MAIN *****/

/***** FUNCTION USR_INTERFACE *****/
/*
/* This function is the main menu system which provides user
/* control to define camera parameters, block structure and motion
/* parameters, output file specification, and initiate simulated
/* imagery generation.
*/

void usr_interface(void)
{
    char ch;                 /*Char read from keyboard      */
    int done;                /*Main loop control           */
    int menu_x = 20, menu_y = 10; /*Menu and title positions    */
    int title_x = 18, title_y = 8;

    done = 0;               /*Initially false             */
    while (!done)           /*Main loop of program       */
    {
        clrscr();           /*Display menu               */
        gotoxy(title_x,title_y);
        cputs("Image Sequence Generation: Planar Motion");
        gotoxy(menu_x,menu_y);
        cputs("F1 - Camera Parameters");
        gotoxy(menu_x,menu_y+1);
        cputs("F2 - Block Structure and Motion Parameters");
        gotoxy(menu_x,menu_y+2);
        if (output_enable)
            printf("F3 - Output Enable (Currently ON -> %s)",filename);
        else
            cputs("F3 - Output Enable (Currently OFF)");
        gotoxy(menu_x,menu_y+3);
        cputs("F10 - Run Imagery Simulation");
        gotoxy(menu_x,menu_y+6);
        cputs("ESC - QUIT");

        ch = getch();       /*Read command               */
    }
}

```

```

if ((!ch) && (kbhit()))      /*Function key requires two reads */
    ch = getch();
switch (ch)                  /*Execute desired function      */
{
    case F1: get_cam_parms(); break;
    case F2: get_block_parms(&block); break;
    case F3: enable_output(); break;
    case F10: generate_imagery(); break;
    case ESC: done = 1; break;
} /*End Switch ch*/
} /*End While !done*/
clrscr();
}
/***** END FUNCTION USR_INTERFACE *****/

/***** FUNCTION GET_CAM_PARMS *****/
/*
/* This function provides access to camera position, orientation
/* and focal length parameters.
*/

void get_cam_parms(void)
{
float rad_2_deg, deg_2_rad;    /*Angular conversion factors */
float temp,temp1,temp2;       /*Temporary variables         */
int done;                     /*Loop control                */
char ch;                      /*Char read from keyboard     */

done = 0;                     /*Initially false              */
rad_2_deg = 180.0/M_PI;
deg_2_rad = M_PI/180.0;
while (!done)                 /*Main loop of function      */
{
    clrscr();
    cputs("Current Camera Parameters:\r\n");
    printf("F1 - Focal Length      = %g m\r\n",cam_parms.F_len);
    printf("F2 - Position (x, y, z) = (%g, %g, %g) m\r\n",
        cam_parms.pasn[0],cam_parms.pasn[1],cam_parms.pasn[2]);
    printf("F3 - Angles (pan, tilt) = (%g, %g) Deg.\r\n",
        rad_2_deg*cam_parms.C_ang[0],rad_2_deg*cam_parms.C_ang[1]);
    cputs("ESC - Exit\r\n");
    cputs("\r\nParameter to change: ");
    ch = getch();              /*Read command                 */
    if ((!ch) && (kbhit()))
        ch = getch();
    cputs("\r\n\r\n");
    switch(ch)                 /*Execute command              */

```

```

    {
        case ESC: done = 1; break;
        case F1: cputs("New Focal Length (m): ");
                scanf("%f",&temp);
                cam_parms.F_len = temp;
                break;
        case F2: cputs("New Position (m) <x y z>: ");
                scanf("%f %f %f",&temp,&temp1,&temp2);
                cam_parms.pasn[0] = temp;
                cam_parms.pasn[1] = temp1;
                cam_parms.pasn[2] = temp2;
                break;
        case F3: cputs("New camera angles (Deg) <pan tilt>: ");
                scanf("%f %f",&temp,&temp1);
                cam_parms.C_ang[0] = deg_2_rad*temp;
                cam_parms.C_ang[1] = deg_2_rad*temp1;
                get_rot(&(cam_parms.Iec[0][0]),
                        &(cam_parms.C_ang[0]));
                transpose(&(cam_parms.Iec[0][0]));
                break;
    } /*End Switch ch*/
} /*End While !done*/
}
/***** END FUNCTION GET_CAM_PARMS *****/

/***** FUNCTION GET_BLOCK_PARMS *****/
/*
/* This function provides access to initial block position and
/* orientation, block structure, and translational and rotational
/* velocities. The time sequence for simulation is also defined
/* through this routine.
*/

void get_block_parms(struct block_type *this_block)
{
float rad_2_deg, deg_2_rad;          /*Angular conversion factors */
float temp,temp1,temp2;              /*Temporary variables */
int done;                            /*Loop control */
int numsteps;                        /*Verify number simulation steps */
char ch,ch1;                         /*Characters read from keyboard */

    done = 0;                          /*Initially false */
    rad_2_deg = 180.0/M_PI;
    deg_2_rad = M_PI/180.0;
    while (!done)                      /*Main loop of function */
    {
        clrscr();

```

```

cputs("Current Block Parameters:\r\n");
printf("F1 - Dimension (depth x) = %g m\r\n",
      this_block->dimension[0]);
printf("      (width y) = %g m\r\n",
      this_block->dimension[1]);
printf("      (height z) = %g m\r\n",
      this_block->dimension[2]);
printf("F2 - Initial Position (x,y) = (%g, %g) m\r\n",
      this_block->posn[0],this_block->posn[1]);
printf("F3 - Initial Yaw Angle = %g Deg.\r\n",
      rad_2_deg*this_block->E_ang[0]);
printf("F4 - Trans. Velocity (vx,vy) = (%g, %g) m/s\r\n",
      this_block->T_vel[0],this_block->T_vel[1]);
printf("F5 - Ang. Velocity (wz) = %g Deg/s\r\n",
      rad_2_deg*this_block->A_vel[2]);
printf("F6 - Time (start,step,stop) = (%g, %g, %g) Sec.\r\n",
      start,step,stop);
if (this_block->occ_flag)
  cputs("F7 - Toggle Occlusion (Currently ON)\r\n");
else
  cputs("F7 - Toggle Occlusion (Currently OFF)\r\n");
cputs("ESC - Exit\r\n");
cputs("\r\nParameter to change: ");
ch = getch(); /*Get command */
if ((!ch) && (kbhit()))
  ch = getch();
cputs("\r\n\r\n");
switch(ch) /*Execute command */
{
  case ESC: done = 1; break;
  case F1: cputs("New Dimensions (m)<depth width height>: ");
    scanf("%f %f %f",&temp,&temp1,&temp2);
    this_block->dimension[0] = temp;
    this_block->dimension[1] = temp1;
    this_block->dimension[2] = temp2;
    break;
  case F2: cputs("New Position (m) <x y>: ");
    scanf("%f %f",&temp,&temp1);
    this_block->posn[0] = temp;
    this_block->posn[1] = temp1;
    break;
  case F3: cputs("New Yaw Angle (Deg) : ");
    scanf("%f",&temp);
    this_block->E_ang[0] = deg_2_rad*temp;
    get_rot(&(this_block->Ioe[0][0]),
            &(this_block->E_ang[0]));

```

```

        break;
    case F4: cputs("New Trans. Velocity (m/s) <vx vy>: ");
             scanf("%f %f",&temp,&temp1);
             this_block->T_vel[0] = temp;
             this_block->T_vel[1] = temp1;
             break;
    case F5: cputs("New Ang. Velocity (Deg/s) <wz>: ");
             scanf("%f",&temp);
             this_block->A_vel[2] = deg_2_rad*temp;
             break;
    case F6: cputs("New time sequence <start step stop>: ");
             scanf("%f %f %f",&start,&step,&stop);
             if ((start > stop) || (step == 0.0))
             {
                 cputs("Data error...Hit any key to continue");
                 getch(); cputs("\r\n");
                 start = 0; stop = 0; step = 1;
             } /*End If*/
             else
             {
                 numsteps = (int) 2.0+(stop-start)/step;
                 printf("Start = %g, Step = %g, ",start,step);
                 printf("Stop = %g, No. Steps = %d\r\n",
                     stop,numsteps);
                 cputs("Is this correct? <y,n> ");
                 if (((ch1 = getch()) != 'y') && (ch1 != 'Y'))
                 {
                     start = 0; stop = 0; step = 1;
                 } /*End IF*/
             } /*End Else */
             break;
    case F7: this_block->occ_flag = !this_block->occ_flag;
             break;
} /*End Switch ch*/
} /*End While 'done*/
}

/***** END FUNCTION GET_BLOCK_PARMS *****/

/***** FUNCTION ENABLE_OUTPUT *****/
/*
/* This function allows the user to run several trials before
/* actually writing data to a file. This function first checks to
/* see if output is already enabled, meaning that no data has yet
/* been written to the file (after a trial run executed with output
/* enabled, the file is automatically closed and this flag reset).
/* If output is currently disabled, an output file is opened and
*/

```

```

/* will be written to during the next trial. */

void enable_output(void)
{
char ch; /*Character from keyboard */

clrscr();
if (output_enable) /*If already enabled... */
{
printf("Output currently enabled to file %s\r\n",filename);
cputs("Further output will be written to this file\r\n");
cputs("Close current file? <y,n> ");
ch = getch(); cputs("\r\n");
if ((ch == 'y') || (ch == 'Y'))
{
fclose(outfile);
output_enable = 0;
} /*End If*/
} /*End If */
else /*If output not enabled... */
{
outfile = open_write_file("Output filename for measurements: ");
if (outfile != NULL)
output_enable = 1; /*Enable if file access successful*/
} /*End Else*/
}
/***** END FUNCTION ENABLE_OUTPUT *****/

/***** FUNCTION GENERATE_IMAGERY *****/
/*
/* This function controls imagery generation. Two graphics pages
/* are employed so that the image of the block can be erased from
/* the second-last frame and drawn according to the current state
/* on the invisible page. During this time, the other page is
/* visible to the user. Once the hidden image is complete, the roles
/* of the graphics pages are reversed. This practice provides a
/* very smooth transition from one image frame to the next. */

void generate_imagery(void)
{
int active_page; /*Active = hidden page */
int numsteps; /*Number of time steps in trial */
int k,i; /*Counters in for loops */
int status_x = 135,status_y = 300; /*Status line posn, graph coords*/
char ch; /*Character from keyboard */
float init_posn[3],init_or[3]; /*Storage for initial block state */

```

```

for (i=0;i<3;i++)          /*Save initial block state      */
{
    init_posn[i] = block.posn[i];
    init_or[i] = block.E_ang[i];
} /*End For*/
active_page = 0;             /*Define initial active page      */
numsteps = (int) 2.0*(stop-start)/step; /*Compute No. time steps */
initialize();               /*Initialize graphics screen    */
setactivepage(active_page); /*Define drawing page          */
draw_block(&block,active_page); /*Draw block at initial state */
settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
outtextxy(status_x,status_y,
           "(ESC TO EXIT, ANY OTHER KEY TO CONTINUE)");
ch = getch();               /*Allow user to abort          */
if (ch == ESC)
{
    clearviewport();
    closegraph();
    return;                  /*EXIT FUNCTION generate_imagery()*/
} /*End If*/
clearviewport();            /*Clear text by clearing viewport */
draw_block(&block,active_page); /*Redraw block at initial state */
if (output_enable)         /*Output measurements if enabled */
    output_measurements(outfile,&block,active_page);

for (k=1; k<numsteps; k++) /*Start trial                  */
{
    active_page = (++active_page)%2; /*Toggle active page to hidden*/
    setactivepage(active_page);
    move_block(&block,step);        /*Compute next block state      */
    draw_block(&block,active_page); /*Draw block on active page     */
    setvisualpage(active_page);     /*Show block at new state       */
    if (output_enable)             /*Output measurements if enabled */
        output_measurements(outfile,&block,active_page);
} /*End For*/
outtextxy(status_x,status_y,
           "(RUN COMPLETE, HIT ANY KEY TO CONTINUE)");
getch();
if (output_enable)          /*Close file and update directory */
{
    fclose(outfile);
    output_enable = 0;
} /*End If*/
clearviewport();            /*Clear graphics                */
closegraph();               /*Return to text mode           */

```



```

    for (i=0;i<3;i++)          /*Restore initial block state */
    {
        block.pasn[i] =init_pasn[i];
        block.E_ang[i] = init_or[i];
    } /*End For */
    get_rot(&(block.Ioe[0][0]),&(block.E_ang[0]));
}
/***** END FUNCTION GENERATE_IMAGERY *****/

/***** FUNCTION OPEN_WRITE_FILE *****/
/*
/* This function opens an ASCII file for output data. The file is
/* checked for prior existence in which case confirmation is
/* required before overwriting takes place. This function returns
/* NULL on failure, and a pointer to a valid file stream otherwise.
/* The input string msg contains the prompt for the filename.
*/

FILE *open_write_file(char *msg)
{
FILE *out;          /*Local stream pointer */
char ch;            /*Character read from keyboard */
int done;           /*Loop control */

    cputs(msg);      /*Display prompt */
    scanf("%s",filename); /*Get filename (max 70 chars) */
    cputs("\r\n");
    done = 0;         /*Initially false */
    while ((!access(filename,0)) && (!done)) /*Check prior existence */
    {
        printf("%s already exists. Overwrite? <y,n,ESC> ",filename);
        ch = getch(); cputs("\r\n");
        if ((ch == 'y') || (ch == 'Y')) /*Overwrite permission granted*/
            done = 1;
        else if (ch == ESC)
            return(NULL); /*EXIT FUNCTION open_write_file() */
        else
        {
            /*Any other character means select*/
            /* different filename */
            cputs(msg);
            scanf("%s",filename);
            cputs("\r\n");
        } /*End Else*/
    } /*End While*/
    if ((out = fopen(filename,"wt")) == NULL) /*Open ASCII file */
    {
        cputs("File open error...Hit any key to continue");
        getch();
    }
}

```

```

    } /*End if*/
    return(out);                                /*Return stream pointer */
}
/***** END FUNCTION OPEN_WRITE_FILE *****/

/***** FUNCTION MOVE_BLOCK *****/
/*
/* This function performs the time update of block position and
/* orientation over a single sample period. Planar motion, and
/* constant translational and rotational velocity are assumed.
*/

void move_block(struct block_type *this_block,float sample_T)
{
    int i,j,k;                                /*Counters */
    float w,temp;                             /*Temporary variables */

    /*Update position -- only x-y coordinates change for planar motion*/
    this_block->posn[0] += sample_T*this_block->T_vel[0];
    this_block->posn[1] += sample_T*this_block->T_vel[1];

    /*Update orientation -- only yaw rates employed for planar motion */
    temp = this_block->E_ang[0];
    temp += sample_T*this_block->A_vel[2]; /*Yaw change only */
    if (temp > M_PI)                       /* limit range to +- PI */
        temp -= 2*M_PI;
    else if (temp < -M_PI)
        temp += 2*M_PI;
    this_block->E_ang[0] = temp;             /*Update block state */
    this_block->E_ang[1] = 0;
    this_block->E_ang[2] = 0;
    get_rot(&(this_block->Ioe[0][0]),&(this_block->E_ang[0]));
}
/***** END FUNCTION MOVE_BLOCK *****/

/***** FUNCTION OUTPUT_MEASUREMENTS *****/
/*
/* This function writes true image plane measurements of all feature*/
/* points (whether or not they are occluded) to the output file. */
/* The kth line of the output file has the format */
/*      u0,v0,u1,v1,...,u7,v7,s0,s2,...,s7 */
/* where (ui,vi) are coordinates of the ith feature point in the */
/* kth image with unity focal length, and si are integer flags which*/
/* are zero if feature point pi is occluded, and one if pi is */
/* visible.
*/

void output_measurements(FILE *outfile,struct block_type *this_block,

```

```

                                int page)
{
int status[8];                  /*Visibility flags          */
int i,k;                        /*Counters              */

    for (i=0;i<8;i++)           /*Reset all flags      */
        status[i] = 0;
    for (i=0;i<6;i++)           /*Loop over 6 faces of block */
    {
        if (this_block->vis_faces[page][i]) /*If visible...      */
        {
            for (k=0;k<4;k++) /*Four feature points/block face */
                status[P_to_F[i][k]] = 1;
        } /*End If*/
    } /*End For*/
    for(i=0;i<8;i++)            /*Output 8 image plane points */
        fprintf(outfile,"%g %g ",
            this_block->cam_vert[0][i],this_block->cam_vert[1][i]);

    for(i=0;i<8;i++)            /*Output 8 visibility flags */
        fprintf(outfile,"%d ",status[i]);

    fprintf(outfile,"\r\n");
}
/***** END FUNCTION OUTPUT_MEASUREMENTS *****/

/***** FUNCTION DEFAULT_INIT *****/
/*
/* This function defines default camera and block parameters and
/* controls the opening demo screen.
*/

void default_init(void)
{
int i,j;                        /*Counters              */
int active_page;               /*Drawing page (hidden) */
float bl_dim[3];               /*Initial dimension of block */
float bl_or[4];                /*Initial orientation    */
float bl_posn[3];              /*Initial position      */
float bl_Tvel[3];              /*Translational velocity */
float bl_Avel[3];              /*Angular velocity       */
float T;                       /*Demo sample period     */
int title1_x = 140, title1_y = 25; /*Demo title positions */
int title2_x = 80, title2_y = 250;
int title3_x = 200, title3_y = 300;

```

```

/* initialize camera parameters */
cam_parms.F_len = 3.0E-3; /* default focal length */
for (i=0;i<3;i++) /*Camera position and orientation */
{
    cam_parms.pasn[i] = 0.0;
    cam_parms.C_ang[i] = 0.0;
} /*End For*/
cam_parms.pasn[2] = -30.0; /* 30 units UP from Fe */
cam_parms.C_ang[1] = -45.0*M_PI/180.0; /* Tilt DOWN 45 degrees */
get_rot(&(cam_parms.Iec[0][0]),&(cam_parms.C_ang[0]));
transpose(&(cam_parms.Iec[0][0])); /*Must transpose: get_rot->Ice*/

/* These parameters taken from data sheets for General Electric */
/* model TN2507 CID video camera. */
/* Image plane width = 8.7841E-3m for GE (6.273E-3 for Sony) */
/* Image plane height = 6.5824E-3m for GE (4.82E-3 for Sony) */
cam_parms.IM_size[0] = 8.7841E-3; /*Width */
cam_parms.IM_size[1] = 6.5824E-3; /*Height */

for (i=0;i<3;i++) /*Default block definition */
{
    bl_or[i] = 0.0; /*initially aligned with Fe */
    bl_pasn[i] = 0.0;
    bl_dim[i] = 10.0;
    bl_Tvel[i] = 0.0; /*Zero translational velocity */
    bl_Avel[i] = 0.0;
}
bl_pasn[0] = 30.0; /*30 m in depth from camera */
bl_pasn[1] = -25.0; /*25 m left of optical axis */
bl_dim[1] = 20.0; /*Twice as wide as tall and deep */
bl_Avel[2] = -1.0; /*Angular velocity */

start = 0; /*Default time interval */
step = 0.1;
stop = 10;

/*Define the default block--with occlusion */
define_block(&block,1,bl_dim,bl_pasn,bl_or,bl_Tvel,bl_Avel,1);

/*Second block for demo screen--without occlusion*/
bl_pasn[1] = 25.0; /*25 m right of optical axis */
define_block(&block2,2,bl_dim,bl_pasn,bl_or,bl_Tvel,bl_Avel,0);

active_page = 0; /*Drawing graphics page */
initialize(); /*Initialize graphics screen */
draw_block(&block,active_page); /*Draw initial blocks */
draw_block(&block2,active_page);

```

```

T = 0.08;                                /*Demo sampling period */

setactivepage(0);                          /*Display titles */
settextstyle(TRIPLEX_FONT,HORIZ_DIR,5);
outtextxy(title1_x,title1_y,"PLANAR MOTION");
outtextxy(title2_x,title2_y,"IMAGERY GENERATION");
settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
outtextxy(title3_x,title3_y,"(HIT ANY KEY TO CONTINUE)");
setactivepage(1);                          /*Must do it on both pages */
settextstyle(TRIPLEX_FONT,HORIZ_DIR,5);
outtextxy(title1_x,title1_y,"PLANAR MOTION");
outtextxy(title2_x,title2_y,"IMAGERY GENERATION");
settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
outtextxy(title3_x,title3_y,"(HIT ANY KEY TO CONTINUE)");
setactivepage(active_page);
while (!kbhit())                          /*Main loop for demo */
{
    active_page = (++active_page)%2;
    setactivepage(active_page);            /*Draw on hidden page */
    move_block(&block,T);                  /*Time update of block states */
    move_block(&block2,T);
    draw_block(&block,active_page); /*Draw both occluded and nonocc.*/
    draw_block(&block2,active_page);
    setvisualpage(active_page);            /*Display new blocks */
} /*End While*/

getch();                                  /*Clear char from input stream */
clearviewport();                          /*Clear graphics page */
closegraph();                             /*Return to text mode */
bl_or[0] = 0.0;                           /*Restore default block */
bl_posn[1] = 0.0;
define_block(&block,1,bl_dim,bl_posn,bl_or,bl_Tvel,bl_Avel,1);
}

/***** END FUNCTION DEFAULT_INIT *****/

/***** FUNCTION INITIALIZE *****/
/*
/* This function initializes the graphics screen and defines scaling
/* parameters. Note that the appropriate *.BGI file must be
/* available even after compilation and link to BL_WORLD.EXE. Here,
/* a Hercules graphics card (HERC.BGI) is assumed, but the program
/* will also run on EGA or VGA (EGAVGA.BGI) systems. The specific
/* system graphics device is auto-detected through standard TURBO C
/* function calls.
*/

void initialize(void)
{

```

```

int i;                                /*Counter                                */
int max_x,max_y;                      /*Max screen dimensions (graphics)*/
int graphdriver = DETECT, graphmode; /*required BGI parameters          */
float temp,pix_y,pix_x;              /*Temporary variables              */
int temp_max_x,temp_max_y;
int aspect_x,aspect_y;               /*Aspect integers                  */
float aspect_rat;                   /*Aspect ratio                     */
int left_scr,top_scr;               /*Boundaries of image on monitor  */
int right_scr,bottom_scr;

initgraph(&graphdriver,&graphmode,""); /*BGI call to init graphics*/
setactivepage(0);                   /*Define drawing and visible page*/
setvisualpage(0);
max_x = getmaxx();                  /*Dimensions of graphics screens */
max_y = getmaxy();
/*All x-screen coords to be multiplied by aspect_rat */
getaspectratio(&aspect_x,&aspect_y);
aspect_rat = (1.0*aspect_y)/(1.0*aspect_x);

/*Determine scaling parameters to visual screen */
temp = aspect_rat*cam_parms.IM_size[0]/cam_parms.IM_size[1];
pix_y = (float) (max_y-2);
pix_x = pix_y*temp;
if (pix_x > (1.0*(max_x-2)))
{
    pix_x = (float) (max_x-2);
    pix_y = pix_x/temp;
} /*End If*/
cam_parms.m_to_scr[0] = pix_x/cam_parms.IM_size[0];
cam_parms.m_to_scr[1] = pix_y/cam_parms.IM_size[1];

/*Redefine max_x and max_y and set view parameters */
temp_max_x = max_x;
temp_max_y = max_y;
max_x = (int) pix_x;
max_y = (int) pix_y;
left_scr = (temp_max_x - max_x)/2;
top_scr = (temp_max_y - max_y)/2;
right_scr = left_scr + max_x + 1;
bottom_scr = top_scr + max_y + 1; /*Draw outline box */
line(left_scr,top_scr,right_scr,top_scr);
line(left_scr,top_scr,left_scr,bottom_scr);
line(right_scr,top_scr,right_scr,bottom_scr);
line(left_scr,bottom_scr,right_scr,bottom_scr);

```

```

    setactivepage(1);          /*Must do on both pages      */
    line(left_scr,top_scr,right_scr,top_scr);
    line(left_scr,top_scr,left_scr,bottom_scr);
    line(right_scr,top_scr,right_scr,bottom_scr);
    line(left_scr,bottom_scr,right_scr,bottom_scr);
    setactivepage(0);

                                /*Set viewport for visual screen */
    setviewport(left_scr+1,top_scr+1,right_scr-1,bottom_scr-1,1);
    CFOV_x = max_x/2;          /*Set CFOV coords          */
    CFOV_y = max_y/2;
}
/***** END FUNCTION INITIALIZE *****/

/***** FUNCTION DEFINE_BLOCK *****/
/*
/* This function initialized block parameters according to received */
/* data.                                                                */

void define_block(struct block_type *this_block,int block_num,
                  float *dimension_0,float *posn_0,float *E_ang_0,
                  float *Tvel_0,float *Avel_0,int occflag)
{
    int i,j,k;                  /*Counters                  */
    float temp;                 /*Temporary variable        */

    this_block->ID_num = block_num; /*Block number (not used here) */
    this_block->occ_flag = occflag; /*1=occlusion, 0 = no occlusion */
    for (i=0;i<3;i++)
    {
        this_block->dimension[i] = dimension_0[i];
        this_block->posn[i] = posn_0[i];
        this_block->E_ang[i] = E_ang_0[i];
        this_block->T_vel[i] = Tvel_0[i];
        this_block->A_vel[i] = Avel_0[i];
    } /*End For*/
    get_rot(&(this_block->Ioe[0][0]),&(this_block->E_ang[0]));

    for (i=0;i<8;i++)           /*Initialize block structure in Fo*/
        for (j=0;j<3;j++)
            this_block->vert[j][i] = dimension_0[j]/2.0;

    for (i=0;i<8;i++)
    {
        if ((i==1) || (i==2) || (i==5) || (i==6))
            this_block->vert[2][i] = -this_block->vert[2][i];
        if (i>3)

```

```

        this_block->vert[1][i] = -this_block->vert[1][i];
    if ((i>1) && (i<6))
        this_block->vert[0][i] = -this_block->vert[0][i];
    } /*End For*/

/* NOTE: This statement places origin of Fo on bottom of block */
for (i=0;i<8;i++)
    this_block->vert[2][i] -= dimension_0[2]/2.0;

for (i=0;i<8;i++) /*Initialize image locations of block vertices*/
{
    this_block->im_vert[0][i][0] = CFOV_x;
    this_block->im_vert[0][i][1] = CFOV_y;
    this_block->im_vert[1][i][0] = CFOV_x;
    this_block->im_vert[1][i][1] = CFOV_y;
} /*End For*/

for (i=0;i<6;i++) /*Define no faces as being visible*/
{
    this_block->vis_faces[0][i] = 0;
    this_block->vis_faces[1][i] = 0;
} /*End For*/
}

/***** END FUNCTION DEFINE_BLOCK *****/

/***** FUNCTION TRANSPOSE *****/
/*
/* This function computes the transpose of the 3x3 matrix R. In all
/* cases here, R is an orthogonal matrix, hence the transpose is the
/* inverse.
*/

void transpose(float *R)
{
    float tmp; /*Temporary variable */

    tmp = R[1]; /*Switch R[0][1] and R[1][0] */
    R[1] = R[3];
    R[3] = tmp;

    tmp = R[2]; /*Switch R[0][2] and R[2][0] */
    R[2] = R[6];
    R[6] = tmp;

    tmp = R[5]; /*Switch R[1][2] and R[2][1] */
    R[5] = R[7];
    R[7] = tmp;
}

/***** END FUNCTION TRANSPOSE *****/

```



```

/***** FUNCTION GET_ROT *****/
/*
/* This function computes the rotation matrix R from Euler angles
/* received in params. Params describes orientation of a reference
/* frame (either Fc or Fo, here) with respect to the earth fixed
/* frame Fe in terms of Euler angles (yaw, pitch, roll). The
/* computed rotation matrix represents the transformation from
/* Fc or Fo to Fe, Ice or Ioe, respectively. In order to obtain Iec,*/
/* function transpose() must be used. */

void get_rot(float *R,float *params)
{
    float c_yaw,s_yaw;          /*Cos and sin of yaw angle      */
    float c_pit,s_pit;          /*Cos and sin of pitch angle   */
    float c_roll,s_roll;        /*Cos and sin of roll angle    */
    float s_roll_s_pit,c_roll_s_pit; /*Products of trig functions  */

                                /*Do all trig functions      */
    c_yaw = cos(params[0]); s_yaw = sin(params[0]);
    c_pit = cos(params[1]); s_pit = sin(params[1]);
    c_roll = cos(params[2]); s_roll = sin(params[2]);
    s_roll_s_pit = s_roll*s_pit;
    c_roll_s_pit = c_roll*s_pit;

    R[0] = c_yaw*c_pit;          /*Compute rotation matrix    */
    R[1] = c_yaw*s_roll_s_pit - s_yaw*c_roll;
    R[2] = s_yaw*s_roll + c_yaw*c_roll_s_pit;
    R[3] = s_yaw*c_pit;
    R[4] = c_yaw*c_roll + s_yaw*s_roll_s_pit;
    R[5] = s_yaw*c_roll_s_pit - c_yaw*s_roll;
    R[6] = -s_pit;
    R[7] = c_pit*s_roll;
    R[8] = c_pit*c_roll;
}
/***** END FUNCTION GET_ROT *****/

/***** FUNCTION DRAW_BLOCK *****/
/*
/* This function first erases the image from the hidden page and
/* then draws the block image at its current state. Since two
/* graphics are employed, it is the image of the second last block
/* state which must be erased. Each feature point lies at the
/* intersection of three planar surfaces (block faces). If all three
/* faces are not visible from the camera position, then that feature
/* point is occluded. Any face is visible from the camera if the

```

```

/* scalar product of the outer normal to the face and a vector from */
/* the centre of projection to any point on that face is negative */
/* (i.e. the cosine of the angle between these vectors is negative).*/
/* Another way to look at the occlusion problem, is that a face of */
/* block is passing through a transition from being visible to being*/
/* occluded, or vs. vs., if the plane defined by this block face */
/* contains the centre of projection (CoP). */

void draw_block(struct block_type *this_block,int page)
{
    float rc_mat[3][8]; /*Vectors from CoP to feature points in Fc */
    float dot_prod[6]; /*Dot (scalar) product discussed above */
    int poly_points[10]; /*Points on image polygon for each face */
    float Ioc[3][3]; /*Transformation from Fo to Fc */
    float Tco[3]; /*Translation from Fc to Fo, in Fc */
    int i,j,k; /*Counters */
    float proj_factor_x,proj_factor_y,temp; /*Projection factors */

    setcolor(0); /*Erase 2nd last drawing from hidden page */
    for (k=0;k<6;k++) /*Loop over all six faces */
    {
        if (!(this_block->occ_flag) || (this_block->vis_faces[page][k]))
        {
            for (i=0;i<5;i++) /*Five points for each poly call */
            {
                poly_points[2*i] =
                    this_block->im_vert[page][P_to_F[k][i]][0];
                poly_points[2*i+1] =
                    this_block->im_vert[page][P_to_F[k][i]][1];
            } /*End For*/
            drawpoly(5,poly_points);
        } /*End If*/
    } /*End For*/
    setcolor(1); /*Set colour to bright */

    for (i=0;i<3;i++) /*Compute Ioc */
    {
        for (j=0;j<3;j++)
        {
            temp = 0.0;
            for (k=0;k<3;k++)
                temp += cam_parms.Iec[i][k]*this_block->Ioe[k][j];
            Ioc[i][j] = temp;
        } /*End For*/
    }

    for (i=0;i<3;i++) /*Compute translation from Fc to Fo in Fc */
        Tco[i] = this_block->posn[i] - cam_parms.posn[i];

```

```

for (i=0;i<3;i++)
{
    /*Load translation into rc matrix      */
    temp = 0.0;
    for (j=0;j<3;j++)
        temp += cam_parms.Iec[i][j]*Tco[j];
    for (k=0;k<8;k++)
        rc_mat[i][k] = temp;
} /*End For*/

for (i=0;i<3;i++) /*Rotate and translate feature points to Fc */
for (j=0;j<8;j++)
{
    temp = rc_mat[i][j];
    for (k=0;k<3;k++)
        temp += Ioc[i][k]*this_block->vert[k][j];
    rc_mat[i][j] = temp;
} /*End For*/

/*Compute projection factors      */
proj_factor_x = cam_parms.F_len*cam_parms.m_to_scr[0];
proj_factor_y = cam_parms.F_len*cam_parms.m_to_scr[1];

for (i=0;i<8;i++) /*Compute eight image plane vertices with unit */
{
    /* focal length      */
    this_block->cam_vert[0][i] = rc_mat[1][i]/rc_mat[0][i];
    this_block->cam_vert[1][i] = rc_mat[2][i]/rc_mat[0][i];
} /*End For*/

for (i=0;i<8;i++) /*Compute eight vertices in visible image*/
{
    this_block->im_vert[page][i][0] =
        (int) (proj_factor_x*rc_mat[1][i]/rc_mat[0][i]) + CFOV_x;
    this_block->im_vert[page][i][1] =
        (int) (proj_factor_y*rc_mat[2][i]/rc_mat[0][i]) + CFOV_y;
} /*End For*/

if (this_block->occ_flag) /*Determine visible faces      */
{
    for (i=0;i<6;i++)
        dot_prod[i] = 0.0;
    for (i=0;i<3;i++)
    {
        dot_prod[0] += Ioc[i][1]*rc_mat[i][0]; /*Face 0      */
        dot_prod[1] -= Ioc[i][0]*rc_mat[i][5]; /*Face 1      */
        dot_prod[2] -= Ioc[i][1]*rc_mat[i][5]; /*Face 2      */
    }
}

```

```

        dot_prod[3] += Ioc[i][0]*rc_mat[i][0]; /*Face 3          */
        dot_prod[4] -= Ioc[i][2]*rc_mat[i][5]; /*Face 4          */
        dot_prod[5] += Ioc[i][2]*rc_mat[i][0]; /*Face 5          */
    } /*End For*/

    for (i=0;i<6;i++)
        if (dot_prod[i] < 0.0)
            this_block->vis_faces[page][i] = 1;
        else
            this_block->vis_faces[page][i] = 0;
    } /*End If*/
else
    for (i=0;i<6;i++)          /*Set all faces visible      */
        this_block->vis_faces[page][i] = 1;

                                /*Draw new block          */
    for (k=0;k<6;k++)          /*Loop over all six faces    */
        if (!(this_block->occ_flag) || (this_block->vis_faces[page][k]))
        {
            for (i=0;i<5;i++)    /*Five points for each poly call */
            {
                poly_points[2*i] =
                    this_block->im_vert[page][P_to_F[k][i]][0];
                poly_points[2*i+1] =
                    this_block->im_vert[page][P_to_F[k][i]][1];
            } /*End For*/
            drawpoly(5,poly_points);
        } /*End If */
    }
/***** END FUNCTION DRAW_BLOCK *****/

/***** END PROGRAM BL_WORLD *****/

```

A.3 Proposed Planar Motion Technique

The following MatLab script file demonstrates a "proof-of-concept" implementation of the proposed technique for planar motion. Only code associated with both motion and structure estimation with local iterations is shown (PMUS.M: Planar Motion with Unknown Structure).

```
%PMUS.M - MatLab script file for simulation of proposed motion
%          and structure estimation under planar motion with occlusion.
%          Local iterations are employed in this version.

clear;                                %Clear all workspace variables
rand('normal');                       %Gaussian random deviate generation
max_MC = 1;                           %No. runs in Monte Carlo simulation
num_alt = 0;                          %Number of non-convergent runs
max_alt = 20;                         %Max No. non-convergent runs allowed
num_match = 4;                       %Number of feature points
max_it = 10;                         %Max iterations per measurement event
threshold = 0.01;                    %Threshold to terminate iterations
sig_z = 0.02;                        %Measurement noise levels

%The following statements load the ASCII file produced by BL_WORLD.C
%and parse this data into a measurement matrix (im_match) and an
%occlusion status matrix (im_stat). The filename must be specified
%by the user through direct editing of this script file.
load im5hvlw5.dta                    %Load imagery and occlusion map
im_match = im5hvlw5(:,1:16);          %Extract ideal measurements
im_stat = im5hvlw5(:,17:24);          %Extract occlusion map

start = 0;                           %Start time
stop = 20;                           %Stop time
T = 0.1;                             %Sample period
t = start:T:stop;                    %Time vector
[garb,numsteps] = size(t);            %Number of steps in run

%System parameters
L = 1;                               %Focal length
%Structure in Fo
r_match = [ 5.0,  5.0,  -5.0, -5.0,  -5.0,  -5.0,  5.0,   5.0;
            10.0, 10.0,  10.0, 10.0, -10.0, -10.0, -10.0, -10.0;
            0.0,-10.0, -10.0, 0.0,   0.0, -10.0, -10.0,  0.0];
%          p1    p2    p3    p4    p5    p6    p7    p8
%          Rotate structure to match what
%          filter should estimate
```

```

gam = atan2(r_match(2,1),r_match(1,1));
rot = [cos(gam),sin(gam),0;
      -sin(gam),cos(gam),0;
       0      ,  0      ,1];
r_match = rot*r_match;

Pmap = [3,7,1,5];           %Mapping from S to r_match
S = [r_match(:,Pmap(1)),r_match(:,Pmap(2)),... %S = structure set
     r_match(:,Pmap(3)),r_match(:,Pmap(4))];

TEC_E = [0;-30;-30];        %Define camera posn Fo wrt Fe in Fe
C_ang = pi*[45;-45]/180.0;  %Camera orientation pan, tilt

c_temp = cos(C_ang(2));     %Compute rotation from Fe to Fc, IEC
s_temp = sin(C_ang(2));
G2_tilt = [c_temp, 0, -s_temp; %Tilt (pitch) rotation
           0,      1,      0;
           s_temp, 0, c_temp];

c_temp = cos(C_ang(1));
s_temp = sin(C_ang(1));
G3_pan = [c_temp,s_temp,0; %Pan (yaw) rotation
          -s_temp,c_temp,0;
          0,0,1];
IEC = G2_tilt*G3_pan;

TEC_C = IEC*TEC_E;          %Compute translation from
                             % Fe to Fc in Fc
                             %Define cross product matrices
kSH = [0,-1,0;              % k_sharp
       1,0,0;
       0,0,0];
jSH = [0,0,1;               % j_sharp
       0,0,0;
       -1,0,0];

A = [0,1,0,0,0,0;          %Define system matrix x_dot = Ax
     0,0,0,0,0,0;
     0,0,0,1,0,0;
     0,0,0,0,0,0;
     0,0,0,0,0,1;
     0,0,0,0,0,0];
exp_AT = expm(A*T);         %State transition matrix
phi = [exp_AT,zeros(6,3*num_match-2);
       zeros(3*num_match-2,6),eye(3*num_match-2)];
                             %True initial state
x0_true = [30;2.5;-25;5;1+gam;-0.5;S(:,1);S(:,2);S(:,3);S(1,4)];

```

```

                                %Initial covariance
std_x0 = [5;3;5;3;0.35;0.35;5*ones(3*num_match-2,1)];
cov = diag(diag(std_x0*std_x0')); %Cov of prior model
                                %Check for divergence
max_error = [30;10;30;10;3;1.5;30*ones(3*num_match-2,1)];

j = 1;
while (j <= max_MC),            %Main loop of Monte Carlo simulation
    filt_alt_flag = 0;          %Reset non_convergence flag

    if (max_MC == 1),           %If not doing Monte-Carlo run

        %Initialization from first frame of measurements

        h_CM = 4.0;             %Estimated mean height of object
        i_star = 1;             %Use pi_star for initial orientation
                                % estimation
        vis_points = 0;         %Count number of visible match points
        for (n=1:num_match),
            if (im_stat(1,Pmap(n))),
                vis_points = vis_points+1;
            end; %If
        end; %For

        u=zeros(2,vis_points);  %Measurements from 1st frame
        meas_num = 0;
        for (n=1:num_match),
            if (im_stat(1,Pmap(n))),
                meas_num = meas_num + 1;
                u(1,meas_num) = im_match(1,2*Pmap(n)-1) + 3*sig_z*rand;
                u(2,meas_num) = im_match(1,2*Pmap(n)) + 3*sig_z*rand;
            end; %If
        end; %For

        ICE = IEC';             %Rotation from Fc to Fe
        rE = zeros(3,vis_points);
        for (n=1:vis_points),    %Estimate pt posns in Fe
            z = [L;u(:,n)]/norm([L;u(:,n)]); %ray from CoP thru image pt
            lambda = (-h_CM - TEC_E(3))/([0,0,1]*ICE*z);
            rE(:,n) = [1,0,0;0,1,0;0,0,0]*(lambda*ICE*z + TEC_E);
        end; %For
        rE_CG = mean(rE');        %Mean est of Fo posn in Fe
        rE_CG = rE_CG';

                                %Est initial orientation
        z = [L;u(:,i_star)]/norm([L;u(:,i_star)]);
        lambda = - TEC_E(3)/([0,0,1]*ICE*z);

```

```

    rE_istar = lambda*ICE*z + TEC_E;           %Est feature posn in Fe
    r0_istar = rE_istar - rE_CG;               %Vector along x0 in Fe
    i0_E = r0_istar/norm(r0_istar);            %Unit vector along x0 in Fe
    psi_0 = atan2(i0_E(2),i0_E(1));            %Initial orientation

                                     %Initialize filter
    xk_0 = [rE_CG(1);0;rE_CG(2);0;psi_0;0;zeros(3*num_match-2,1)];
else
                                     %Filter IC sample from N(x0_true,cov)
    xk_0 = x0_true + std_x0 .* rand(6+3*num_match-2,1);
end;

x_true = x0_true;                      %Initialize true states
xk = xk_0;                             %Initialize filter state
                                     %Initial measurement step
vis_points = 0;                        %Count number visible match points
for (n=1:num_match),
    if (im_stat(1,Pmap(n))),
        vis_points = vis_points+1;
    end; %If
end; %For

z=zeros(2*vis_points,1);               %Take measurements
meas_num = 0;
for (n=1:num_match),
    if (im_stat(1,Pmap(n))),
        meas_num = meas_num + 1;
        z(meas_num) = im_match(1,2*Pmap(n)-1) + sig_z*rand;
        meas_num = meas_num + 1;
        z(meas_num) = im_match(1,2*Pmap(n)) + sig_z*rand;
    end; %If
end; %For

R = sig_z*sig_z*eye(2*vis_points); %Compute meas Cov R

x_old = xk;                            %Prepare for iteration
done = 0;
i = 1;
while ((~done) & (i <= max_it)),
    x_prev = xk;                        %Save previous iterate

    %Collect true and estimated match points (last two are zero)
    ro_hat = [xk(7:9),xk(10:12),xk(13:15),[xk(16);0.0;0.0]];

    c_hat = cos(xk(5));                 %Compute l0e from estimate
    s_hat = sin(xk(5));

```



```

Ioe_hat = [c_hat,-s_hat,0;s_hat,c_hat,0;0,0,1];

re_hat = Ioe_hat*ro_hat;      %Rotate to Fe
for (n=1:num_match),          % and translate
    re_hat(:,n) = re_hat(:,n) + [xk(1);xk(3);0.0];
end;

rc_hat = IEC*re_hat;          %Rotate to Fc
for (n=1:num_match),          % and translate
    rc_hat(:,n) = rc_hat(:,n) - TEC_C;
end;

nu = zeros(2*vis_points,1);    %Setup innovation nu and H
H = zeros(2*vis_points,6+3*num_match);

meas_num = 0;                  %Compute nu and H
for (n=1:num_match),
    if (im_stat(1,Pmap(n))),
        meas_num = meas_num + 1;

        %Compute innovation for x-measurement
        nu(meas_num) = z(meas_num) - L*rc_hat(2,n)/rc_hat(1,n);

        %Compute linear approximation Hk for x-measurement
        sqr_x = rc_hat(1,n)*rc_hat(1,n);
        temp1 = L*IEC/sqr_x;
        temp2 = temp1*kSH*Ioe_hat*ro_hat(:,n);
        H(meas_num,1) = rc_hat(1,n)*temp1(2,1)-rc_hat(2,n)*temp1(1,1);
        H(meas_num,3) = rc_hat(1,n)*temp1(2,2)-rc_hat(2,n)*temp1(1,2);
        H(meas_num,5) = -rc_hat(:,n)'*kSH*temp2;
        temp3 = -rc_hat(:,n)'*kSH*temp1*Ioe_hat;
        H(meas_num,6+3*n-2) = temp3(1);
        H(meas_num,6+3*n-1) = temp3(2);
        H(meas_num,6+3*n) = temp3(3);

        meas_num = meas_num + 1;

        %Compute innovation for y-measurement
        nu(meas_num) = z(meas_num) - L*rc_hat(3,n)/rc_hat(1,n);

        %Compute linear approximation Hk for y-measurement
        H(meas_num,1) = rc_hat(1,n)*temp1(3,1)-rc_hat(3,n)*temp1(1,1);
        H(meas_num,3) = rc_hat(1,n)*temp1(3,2)-rc_hat(3,n)*temp1(1,2);
        H(meas_num,5) = rc_hat(:,n)'*jSH*temp2;
        temp3 = rc_hat(:,n)'*jSH*temp1*Ioe_hat;
        H(meas_num,6+3*n-2) = temp3(1);

```

```

    H(meas_num,6+3*n-1) = temp3(2);
    H(meas_num,6+3*n) = temp3(3);

    end; %If
end; %For

%Remove last two columns from H (known states)
H = H(:,1:6+3*num_match-2);

Kk = cov*H'*inv(H*cov*H' + R); %Compute Kalman gain
if (max_it > 1) %State update
    xk = x_old + Kk*(nu - H*(x_old - xk));
else
    xk = xk + Kk*nu;
end;

i=i+1;
if (max(abs(x_prev-xk)) < threshold),
    done = 1;
end; %If
end; %While
cov = (eye(6+3*num_match-2) - Kk*H)*cov; %Update covariance

error_hist = xk - x_true; %Store errors

%Compute rotation matrix convergence performance measure
c_true = cos(x_true(5));
s_true = sin(x_true(5));
Ioe_true = [c_true,-s_true,0;s_true,c_true,0;0,0,1];
Ioc_true = IEC*Ioe_true;
Ioc_hat = IEC*Ioe_hat;
J = sqrt(trace((Ioc_hat-Ioc_true)'*(Ioc_hat-Ioc_true)));
J_hist = J;

for k = 2:numsteps, %Main simulation loop
    disp([k,j,num_alt]); %Feedback to user
    x_true = phi*x_true; %True state time update

    xk = phi*xk; %Filter time propagation
    cov = phi*cov*phi'; %Compute prediction covariance

    vis_points = 0; %Count visible feature points
    for (n=1:num_match),
        if (im_stat(k,Pmap(n))),
            vis_points = vis_points+1;
        end; %If
    end; %If
end; %For

```

```

end; %For

z=zeros(2*vis_points,1);      %Take measurements
meas_num = 0;
for (n=1:num_match),
    if (im_stat(k,Pmap(n))),
        meas_num = meas_num + 1;
        z(meas_num) = im_match(k,2*Pmap(n)-1) + sig_z*rand;
        meas_num = meas_num + 1;
        z(meas_num) = im_match(k,2*Pmap(n)) + sig_z*rand;
    end; %If
end; %For

R = sig_z*sig_z*eye(2*vis_points); %Compute meas Cov R

x_old = xk;                    %Prepare for iteration
done = 0;
i = 1;
while ((~done) & (i <= max_it)),
    x_prev = xk;

    %collect true and estimated match points
    ro_hat = [xk(7:9),xk(10:12),xk(13:15),[xk(16);0.0;0.0]];

    c_hat = cos(xk(5));        %Compute Ioe from estimate
    s_hat = sin(xk(5));
    Ioe_hat = [c_hat,-s_hat,0;s_hat,c_hat,0;0,0,1];

    re_hat = Ioe_hat*ro_hat;    %Rotate to Fe
    for (n=1:num_match),        % and translate
        re_hat(:,n) = re_hat(:,n) + [xk(1);xk(3);0.0];
    end;

    rc_hat = IEC*re_hat;        %Rotate to Fc
    for (n=1:num_match),        % and translate
        rc_hat(:,n) = rc_hat(:,n) - TEC_C;
    end;

    nu = zeros(2*vis_points,1); %Setup innovation nu and H
    H = zeros(2*vis_points,6+3*num_match);

    meas_num = 0;                %Compute nu and H
    for (n=1:num_match),
        if (im_stat(k,Pmap(n))),
            meas_num = meas_num + 1;

```

```

%Compute innovation for x-measurement
nu(meas_num) = z(meas_num) - L*rc_hat(2,n)/rc_hat(1,n);

%Compute linear approximation Hk for x-measurement
sqr_x = rc_hat(1,n)*rc_hat(1,n);
temp1 = L*IEC/sqr_x;
temp2 = temp1*kSH*Io_hat*ro_hat(:,n);
H(meas_num,1) = rc_hat(1,n)*temp1(2,1)-rc_hat(2,n)*temp1(1,1);
H(meas_num,3) = rc_hat(1,n)*temp1(2,2)-rc_hat(2,n)*temp1(1,2);
H(meas_num,5) = -rc_hat(:,n)'*kSH*temp2;
temp3 = -rc_hat(:,n)'*kSH'*temp1*Io_hat;
H(meas_num,6+3*n-2) = temp3(1);
H(meas_num,6+3*n-1) = temp3(2);
H(meas_num,6+3*n) = temp3(3);

meas_num = meas_num + 1;

%Compute innovation for y-measurement
nu(meas_num) = z(meas_num) - L*rc_hat(3,n)/rc_hat(1,n);

%Compute linear approximation Hk for y-measurement
H(meas_num,1) = rc_hat(1,n)*temp1(3,1)-rc_hat(3,n)*temp1(1,1);
H(meas_num,3) = rc_hat(1,n)*temp1(3,2)-rc_hat(3,n)*temp1(1,2);
H(meas_num,5) = rc_hat(:,n)'*jSH*temp2;
temp3 = rc_hat(:,n)'*jSH*temp1*Io_hat;
H(meas_num,6+3*n-2) = temp3(1);
H(meas_num,6+3*n-1) = temp3(2);
H(meas_num,6+3*n) = temp3(3);
end; %If
end; %For

%Remove last two columns from H (known states)
H = H(:,1:6+3*num_match-2);
Kk = cov*H'*inv(H*cov*H' + R); %Compute Kalman gain
if (max_it > 1) %State update
    xk = x_old + Kk*(nu - H*(x_old - xk));
else
    xk = xk + Kk*nu;
end;

i=i+1;
if (max(abs(x_prev-xk)) < threshold),
    done = 1;
end; %If
end; %While
cov = (eye(6+3*num_match-2) - Kk*H)*cov; %Update covariance

```

```

error = xk-x_true;           %Store errors
error_hist = [error_hist,error];

%Compute rotation matrix convergence performance measure
c_true = cos(x_true(5));
s_true = sin(x_true(5));
Ioe_true = [c_true,-s_true,0;s_true,c_true,0;0,0,1];
Ioc_true = IEC*Ioe_true;
Ioc_hat = IEC*Ioe_hat;
J = sqrt(trace((Ioc_hat-Ioc_true)'*(Ioc_hat-Ioc_true)));
J_hist = [J_hist,J];

error = abs(error);          %Check for apparent divergence
if ((max_MC > 1) & (k > numsteps/2) & (max(error-max_error) > 0.0)),
    filt_alt_flag = 1;
    num_alt = num_alt + 1;
    j = j - 1;
    disp('Filter did not converge to true parameters.');
```

break;

```

end; %If
end; %For k

if ((max_MC > 1) & (~filt_alt_flag)), %Update statistics of run
    if (j == 1),
        mean_hist = error_hist/max_MC;
        mean_J = J_hist/max_MC;
        sum_sqr = (error_hist .* error_hist)/max_MC;
        sqr_J = (J_hist .* J_hist)/max_MC;
    else
        mean_hist = mean_hist + error_hist/max_MC;
        mean_J = mean_J + J_hist/max_MC;
        sum_sqr = sum_sqr + (error_hist .* error_hist)/max_MC;
        sqr_J = sqr_J + (J_hist .* J_hist)/max_MC;
    end; %Else
end; %If

if (num_alt > max_alt),
    disp('Maximum number of non-convergent runs exceeded!');
    break;
end; %If
j = j+1;
end; %While j<=max_MC loop

%***** END PMUS.M *****
```

END

03·03·92

FIN