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.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

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

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.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, tests publiés, etc.) ne sont pas microfilmés.

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

NL-339 (r. 88/04)

Canada

THE UNIVERSITY OF ALBERTA

IRLM: An Intensity Image System for 3D Object Recognition, Localization, and Motion recovery

BY

©

Jason Jenchang Wu

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH IN PARTIAL
FULFILMENT OF THE REQUIREMENT FOR THE DEGREE OF DOCTOR OF PHILOSOPHY.

DEPARTMENT OF ELECTRICAL ENGINEERING

EDMONTON, ALBERTA

FALL 1988

Permission has been granted to the National Library of Canada to microfilm this thesis and to lend or sell copies of the film.

The author (copyright owner) has reserved other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without his/her written permission.

L'autorisation a été accordée à la Bibliothèque nationale du Canada de microfilmer cette thèse et de prêter ou de vendre des exemplaires du film.

L'auteur (titulaire du droit d'auteur) se réserve les autres droits de publication; ni la thèse ni de longs extraits de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation écrite.

ISBN 0-315-45671-X

THE UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR: Jason Jenchang Wu

TITILE OF THESIS: IRLM: An Intensity Image System for 3D Object Recognition, Localization, and Motion recovery

DEGREE: Ph.D.

YEAR THIS DEGREE GRANTED: 1988

Permission is hereby granted to THE UNIVERSITY of ALBERTA LIBRARY to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purpose only.

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

Jason Jenchang Wu

Jason Wu

Permenent address:

No. 1006, Fonlin 2nd Rd. Daliao district,
Kaohsiung County, TAIWAN, ROC

Date: June 24, 1988

THE UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled, "IRIM: An Intensity Image System for 3D Object recognition, Localization, and Motion Recovery", submitted by Jason Jenchang Wu in partial fulfilment of the requirements for the degree of Doctor of Philosophy in Electrical Engineering.

V. G. Gourishankar

R. E. Rink

Z. J. Koles

Date: June 24, 1988

Abstract

The use of conic primitives in three-dimensional (3D) object recognition, localization, and motion recovery is studied in this thesis. The major result is the closed-form solution for 3D localization problem from different conic correspondences. Based on this result, a "locating-labelling" method is proposed for designing an efficient 3D object recognition system from a single (2D) intensity image. The efficiency of the method comes from the ability to locate an hypothesized object, in 3D, early in the recognition process; that is, its ability to use a minimum number of hypothesized correspondence to locate the object. Once the hypothesized object is located, its validity can be tested immediately. Consequently, the time spent in the recogniton process is mainly for the initial formulation of the minimum number of hypothesized matches; the minimum number of elliptic/hyperbolic and line/point correspondences required is one and three, respectively. Further time reduction is obtained by exploiting the invariant properties and relations of detected image primitives.

Upon completion of recognition of an 3D object from an intensity image, the 3D location of the object is accurately determined by using all established matches. Further, 3D motion of the object from the sample instant of the previous image is recovered by using an extended Kalman Filter algorithm (EKF). The recovered motion can also be used to aid the recognition process. Implementation with real images shows that the recovered location and motion are very accurate (down to 1.0 mm in position error), and recognition of 3D object from 2D images is reliable and economical.

Acknowledgement

The author would like to thank Drs. Gourishankar and Rink of the Department of Electrical Engineering and Dr. Caelli, Killiam Professor in the Department of Psychology for their guidance and support through the my entire Ph.D program. The author would also like to express his gratitude to Drs. Davis, Koles, Lawrence, and Lawson for their valuable comments and suggestions. The author thanks Georgina Burstow for her kindness and help during his stay at the Psycology Department of this University, and to thank John Chen for proof-reading the thesis.

Table of Contents

Chapter

Page

1. Introduction 1

    1.1 Constraints: Expressions of Cues for Perception 3

    1.2 Approaches to Object recognition 13

    1.3 The Approach used in this Thesis 22

2. Object Recognition 27

    2.1 Object Models and Database 29

    2.2 Primitive Detection and Feature Grouping 33

    2.3 Matching Image Primitives and Model Primitives 45

3. Object Localization 58

    3.1 One-Ellipse Localization (The E1 problem) 61

    3.2 n-Ellipse Localization (The En problem, where n > 1) 62

    3.3 Three-Line Localization (The L3 problem) 63

    3.4 n-Line Localization (The Ln problem, where n > 3) 65

    3.5 Three-Point Localization (The P3 problem) 66

    3.6 n-Point Localization (The Pn problem, where n > 3) 67

    3.7 Two-Line-One-Point Localization (The L2P1 problem) 67

    3.8 One-Line-Two-Point Localization (The L1P2 problem) 68

    3.9 k-Line-n-Point Localization (The LkPn problem, where k+n>3) · 68

    3.10 j-Ellipse-k-Line-n-Point Localization (The EjLkPn problem) · 69

4. Estimation of Motion Dynamics Using an Extended Kalman Filter ... 70

    4.1 The Extended Kalman Filter (EKF) 70

    4.2 The Dynamic Model 75

    4.3 The Measurement Model 81

5. Experimental Results..........86

5.1 Camera Calibration..........86

5.2 Object Recognition and Localization..........94

5.3 Motion Recovery..........101

6. Conclusions..........109

References..........117

Appendix A. Perspective Projection of Conic Sections..........124

A.1 Perspective Projection of an Ellipse..........124

A.2 Perspective Projection of a Hyperbola..........126

A.3 Perspective Projection of a Parabola..........127

Appendix B. Finding Constraints for Constraint Propagation..........128

B.1 Possible Quadratic Constraints..........129

B.2 Constraints Propagation in Bottom-Up Matching..........131

B.3 Constraint Propagation in Top-down Matching..........131

Appendix C. Propagating Quadratic Constraints..........133

C.1 Evaluating Intervals of Variables Using Given Quadratic Constraints..........133

C.2 Evaluating Quadratic Expression Using Given Variable Intervals..........136

Appendix D Minimum-Correspondence Localization..........138

D.1 Localization Using One Ellipse Correspondence..........138

D.2 Localization Using Three Line Correspondences..........141

D.3. Localization Using Three Point Correspondences..........145

Appendix E. Extended Kalman Filter (EKF)..........150

Appendix F. Camera Calibraton..........154

List of Tables

Table Page

2.1 Reliability of Measured Attributes for Conic Pimitives 39

2.2 Significance of Detected Conic Primitives 39

2.3 Perspective projection of conic sections 48

4.1 Nomenclature used in the EKF formulation 72

5.1 The trajectory of the programmed motion in the motion recovery experiment 101

List of Figures

Figure Page

1.1 Constraints representing relations of constituents··················6

1.2 An example of inter-relations and intra-relations······················6

1.3 An example of Sementic net···8

1.4 Block diagram of the proposed method for 3D recognition··········24

1.5 Illustration of perspective projection···26

2.1 An example of index table accessed with index feature groups·····32

2.2 Cell structures for detected image points, lines, and conics·····41

2.3 An example of conic cell table··42

2.4 Block diagram of minimum-correspondence hypothesis formulator····52

2.5 Block diagram of hypothesis tester··56

2.6 Illustration of area fit···57

3.1 An illustration for perspective projection of an ellipse·········62

4.1 Block diagram of motion recovery using EKF····································71

5.1 The calibration points on the calibration plate·························90

5.2 An illustration of the motion-generating device························91

5.3 One view of of the motion-generating device····························92

5.4 The sequence of images used in camera calibration··················93

5.5 One view of the wireframe model of the audio cassette············97

5.6 The object coordinate basis in the audio cassette····················98

5.7 The images used in the object recognition experiment·············99

5.8 Some snapshots of the hypotheis testing cycle in the object recognition experiments··100

5.9 The sequence of images used in the motion recovery experiment···106

5.10 The esimation errors in the motion recovery experiment·········107

6.1 3D object recognition using locating-labelling paradigm with 2D sketch and other methods with $2\frac{1}{2}$D sketch......112
6.2 3D object recognition using locating-labelling paradigm with 2D sketch and with $2\frac{1}{2}$D sketch......113

Chapter 1. Introduction

One goal of computer vision research is to design a machine which can recognize three-dimensional (3D) objects from sensed data, determine the locations of these objects relative to the sensor, and predict their motions relative to the sensor. A machine with these capabilities can be of use in factory automation, in home robotics, in underwater or outerspace exploration. In factory automation, the most common task in manufacturing, transportation, storage, and assembly is the recognition and manipulation of workpieces on conveyors or in bins (Rosen, 1979). Since workpieces can overlap one another, the machine must be able to recognize them when they are not spatially separated. In manipulating a workpiece, its stable pose and its location relative to the manipulator have to be determined in order that the manipulator can grasp and move it. Also, a workpiece may be moving relative to the viewing camera or the camera is moving with the manipulator, hence motion information is needed in a dynamic environment. In addition to recognition and manipulation of workpieces, other tasks in factory automation requiring machine vison are visual feedback (of relative location or motion) for manipulator control and workpiece inspection (of missing parts or labels) for quality control.

Responding to these needs, much research has been devoted to the study of 3D object recognition and localization (for a survey, see Besl and Jain, 1985; Chin and Dyer, 1986). However, reliable, efficient, and economical methods for designing such systems have yet to be devised. In the past 20 years, most research on object recognition has used intensity images. Although recognition of 2D objects from a 2D image (e.g., flat objects parallel to the image plane) has been successful in some devised systems, few are capable of recognizing 3D objects from a single intensity image without human intervention (humans are usually able to perform such tasks effortlessly). Faced with the difficulty in using intensity images, range images (either from a range finder or shape-from-X techniques on intensity images) are commonly used in most recent research on object recognition in the hope that 3D recognition from range images will be much easier than from intensity images. Despite the explicit depth information in range images, the difficulty in 3D object recognition remains, as exemplified by the complexity of the methods of Oshima and Shirai (1983), Bolles and Horaud (1984), Grimson and Lozano-Pérez (1985), Faugeras and Ayache (1986), and Pentland (1987).

In this dissertation, a method is proposed to perform 3D object recognition from a single and/or multiple 2D images. A prototype system was built to show the reliability, efficiency, and economy of the proposed method. The efficiency of the proposed method comes from extensive use of location information in the recognition process. The location information about the object to be recognized can be obtained from a minimum number of hypothesized matches between image and model primitives. The minimum number of matches can be as low as one (elliptical correspondence) or as high as three (line or point correspondences). To obtain the location information with minimum computational effort, closed-form solutions are derived for various types of matches. After location is recovered for the hypothesized matches, the matches are tested to determine if they are consistent with the recovered location. If they are consistent, more matches can be hypothesized and location information can be refined by using the augmented set of hypothesized matches, and the augmented matches are tested again with the refined location information. However, if the hypothesized matches are not consistent with the recovered location information, they are rejected, thus leading to a great time saving by avoiding any hypothesis including the rejected matches. As a result, the proposed method can efficiently recognize 3D objects and recover their 3D locations (each with 3 orientation and 3 position parameters) from a single intensity image.

In addition to the recognition and localization capabilities, an Extended Kalman Filter (EKF) approach is used to recover the motion of an object appearing in a sequence of images. The information recovered is the smoothed motion from the sequence of images, which is more reliable than the incremental motion from the direct difference of locations recovered by the recognition process in two consecutive mes.

Prior to introducing the proposed method for 3D recognition, localization, and motion recovery, the remainder of this chapter will provide some basic materials in object recognition and relate the proposed method to previous research.

1.1 Constraints: Expressions of Cues for Perception

Given an intensity image or 2D pattern of illumination, how do we possibly choose one out of a large number of valid 3D interpretations to fit the given image? Studies in biological vision tell us that perception relies heavily on implicit information present in the image.

This information can be perspective, binocular disparity, relative size, relative brightness, shadows and shading, motion parallax (the relative movement of objects at different distance from the viewer), texture gradients, colour, contour, Gestalt groupings, etc. This kind of information provides cues for depth, size, and shape perception, thus imposing constraints on the plausibility of interpretations of the image. In short, such constraints can be used to overcome the essentially underconstrained nature of object recognition from 2D views.

Motivated by the cues used in biological vison, computer vison researchers usually use constraints to express cues (either domain-specific or domain-independent) to solve various vision problems. The trend of using constraints can be traced back to early research on vision, notably Robert's block world (1965), Guzman's SEE (1968), and Waltz's filtering algorithm (Waltz, 1975). For example, a three-line vertex surrounded by regular polygons is good evidence for a cube (Robert, 1965), an arrow vertex of three lines is a good evidence that two regions, defined by the three lines, are linked (Guzman, 1968). Although recent research in computer vision has shifted to early vision such as shape-from-X problem (epitomized by Marr, 1982), the use of constraints in solving vision problem is still prevalent. For example, in shape-from-shading (or surface orientation from image intensity), constraints are the expressions of assumptions such as surfaces are globally continuous and albedo (surface reflectance) is constant or piecewise constant (Brooks and Horn, 1985; Bischof and Ferraro, 1987). The surface continuity assumption relates the surface orientation at a given point to the surface orientation at neighboring points, thus imposing constraints on possible orientation at the given point. The piecewise-constant albedo assumption requires that the points in a certain region should have the same reflectance property. As a second example of using constraints in early vision, consider the approach to stereopsis (depth from two images at different view points) by Marr and Poggio (1976). In their approach, compatibility, uniqueness, and continuity constraints are used to match two images and measure disparity (angular discrepancy in position of the object in two images) between them. The compatibility constraint requires that two primitives in different images match to each other only if they are of the same type. The uniqueness constraint means that each primitive in one image can match only one item from the other image. The continuity constraints requires that the measured disparity between two images varies smoothly.

Since constraints are ubiquitous in the computational approach to vision, it is useful to examine questions as to what constitutes a constraint, in what form a constraint can be represented, and what are the ways to classify constraints? Examining these questions not only will help us analyze previous research, but will also help reveal important ingredients for certain vision problems.

What constitutes a constraint? A constraint describes a property of a constituent or a relation among the constituents in an image or object model, or describes a correspondence among properties/relations in different images or models. Figure 1.1 illustrates properties and relations (intra-relation) of constituents as well as correspondences (inter-relation) of properties and relations. The constituents can be as small as picture elements (pixels) or as large as parts of the object. An example expressing properties, intra-relations, and inter-relation is given in Figure 1.2 by using a coffee mug as the object.

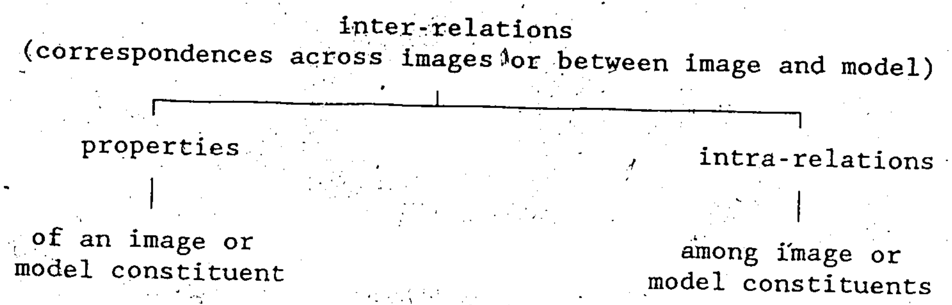
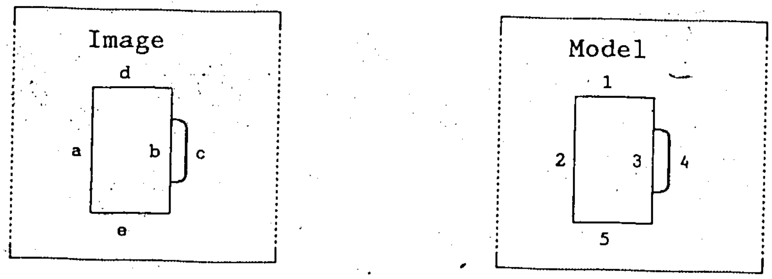

inter-relations
(correspondences across images or between image and model)
properties
intra-relations
of an image or model constituent
among image or model constituents

Figure 1.1 constraints as relations and properties of constituents or as correspondences of relations and properties of constituents.



(a) (b)

Properties on image	Correspondence between properties	Properties in model
ellipse(d)	⟷	ellipse(1)
ellipse(e)	⟷	ellipse(5)
part-ellipse(c)	⟷	part-ellipse(4)
line(a)	⟷	line(2)
line(b)	⟷	line(3)
Relations on image	**Correspondence between relations**	**Relations in model**
parallel(a,b)	⟷	parallel(2,3)
intersect(a,d)	⟷	intersect(2,1)
intersect(b,d)	⟷	intersect(3,1)

(c)

Figure 1.2 . A "coffee mug" example for intra-relations and inter-realtions. The constituents used here are conic primitives. (a) Constituents of image cup. (b) Constituents of model cup. (c) The properties, relations (or intra-relations) and correpondences (or inter-relations).

How to express Constraints. A constraint for a relation or property can be expressed symbolically or numerically. In numerical expression, a constraint is described by an equality/inequality equation. An equality constraint means that attributes of constituents, when combined with a specific rule for expressing a relation or property, must be equal to a certain value. For example, the coordinate transformation between two parts in a rigid object is equal to the combined transformation from the first part through a reference coordinate to the second part. An inequality constraint means that the value from the combination of constituent attributes is within a specified range. As an example in perspective projection with unit focal length, if a point with the relative coordinates (x,y,z) to the view point appears in the perspective image, then the ratios, -x/z and -y/z, correspond to a projection point in the image plane. In other words, the ratios, -x/z and -y/z, must be within the horizontal and vertical limits of the image, respectively.

In the symbolic expression, a relation is abstracted into a terse format such as the first-order predicates and semantic nets. In artificial intelligence terminology, symbolic constraints are usually called production rules (in rule-based systems or production systems) or inference rules (in planning or logic inference). The symbolic expression of relations can be seen from the example of expressing the sentence, "All bridges span roads or rivers". In predicate logic, the sentence can be expressed as:

$$\forall x \{ \mathrm{bridge}(x) \Rightarrow \exists y \{ \{ \mathrm{Road}(y) \vee \mathrm{River}(y) \} \wedge \mathrm{Over}(x,y) \} \}$$

where $\forall$ is the universal quantifier, and $\exists$ is the existential quantifier. $\vee$ is the disjunctive connective, $\wedge$ is the conjunctive connective, and ⇒ is the implication connective.

Still, the same sentence can be expressed in semantic net as the one shown in Figure 1.3. An example of how sementic nets can be used to describe scenes and compare descriptions is the work reported by Winston (1975).

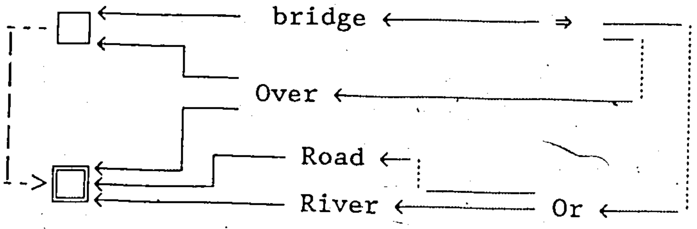


Figure 1.3 A semantic net representation of "All bridges span road or river". ☐ is the universal quantifier, ▣ is the existential quantifier, and ⇒ is the implication symbol. ——→ is the logical link, ——→ is the propositional link, and ----> is the dependence link. This expression follows Schubert's representation of semantic net (1976).

<u>Taxonomy</u> <u>of</u> <u>Constraints</u>. Having defined what constitutes a constraint and its expression types, we can classify the constraints into different classes. The constraints used in vision research can be divided into two categories: model-independent and model-specific. The model-independent constraints are those relations that can be used without resorting to explicit knowledge of the object being viewed. One extreme example of model-independent constraints are the surface geometric properties which are invariant under transformation or projection. For instance, the Gaussian and mean curvatures of surfaces are independent of transformation (Besl and Jain, 1986). The Gaussian and mean curvatures can be derived directly from range images, and

their signs are used to infer the type of viewed surface; mely, peak surface, flat surface, pit surface, minimal surface, ridge surface, saddle ridge, valley surface, or saddle valley (Besl and Jain, 1986). Other model-independent constraints can take the form of assumptions for a specific application domain. An example is the assumption of surface smoothness used in many shape-from-shading formulations.

Model-dependent constraints are those relations imposed by specific objects being viewed with prior world or object knowledge. An example is the restricted range of angle and distance of any two normals of points on two specific surfaces (Grimson and Lozano-Pérez, 1984 and 1985).

Model-independent constraints are commonly used in early (or low level) visual functions to extract higher level descriptions of shape in the image. In most cases, the applications of this kind of constraints is data-driven, although cooperative (or interacting) processes among constituents and coarse-to-fine modular processes can be used. The bottom-up application of model-independent constraints can be seen in such examples as Marr and Poggios's stereopsis (1982), Guzman's SEE (1968), and Waltz's filtering algorithm (1975). However, this is not to say that model-independent constraints can only be used in low level vision. The application of such model-independent constraints can provide valuable guidance in high level visual interpretation, as shown in the use of invariances by Besl and Jain (1986) to help matching model surfaces to a given input image.

On the other hand, model-dependent constraints can only be used in high level vision, because explicit knowledge of the object model is needed. In general, model-based vision systems, such as those by Brooks

(1981), Lee and Fu (1983), Oshima and Shirai (1983), Grimson and Lozano-Pérez (1984,1985), Bolles and Horaud (1984), Faugeras and Hebert (1986), Lowe (1987), and Pentland (1987), all exploit model-specific knowledge to recognize object instances in a scene.

Another classification of constraint types comes from the components of the image formation process. Here the components correspond to object shape and surface albedo, transformations between the objects and view coordinates, projection geometry, and illumination. Object-induced constraints are the properties and relations encoded in the object's surface geometry and its material. These properties and relations are preserved under projection. The object-induced constraints can be model-independent (e.g., the Gaussian curvature) or model-dependent (e.g., the angle between two specific surfaces in the model). In some applications, the constraints can be an assumption or hypothesis; e.g., in shape-from-shading techniques, piece-wise surface continuity and piece-wise constant surface albedo are assumed in order to recover surface parameter and albedo from an intensity image. Here, the constraint is that neighboring pixels in a region on the image should have the same albedo and be on a smooth surface except for those pixels on the contour of a region.

Transformation-induced constraints have two properties: common fate property and interposition property. The first indicates that a transformation of the object should have the same effect on all parts of the object. One extreme of the common fate property is that, if the object is rigid, all parts should comply with the same transformation. The interposition property indicates whether certain parts in an object will be occluded by the object or other objects, provided that the transformations of all objects in the scene are known. Conversely, the interposition constraint means that, if a part of an object is not visible in the scene, then the transformation of the part, when found, should comply with the fact that the part is occluded or outside the field of view.

Projection-induced constraints relate the object and its image through a specific type of projection and its parameters (e.g., the scale of size). Under a certain type of projection, properties and relations of parts in the scene are mapped into certain properties and relations of the corresponding parts in the image. For intensity images, the projection maps relative positions ~~of~~ points in the scene into relative pixels in the image, and the value of each pixel captures the reflectance information of each corresponding spot in the scene. Like intensity images, the range image encodes the relative positions of points in the scene into the relative pixel coordinates in the image; however, the value of each pixel contains the relative distances of the corresponding point in the scene to the view point instead of reflectance information.

Illumination is a factor, in addition to shape and albedo, which affects the intensity value and shading in intensity images. Further, the illuminating light, along with the interposition of parts, produces shadows in the intensity image. One way to use illumination for object recognition is the application of structured light, in which known light sources are used to illuminate the object and the object is viewed by the camera to detecte the object and reconstruct the object properties (Holland, Rossel, and Ward, 1979). In this manner, the structured light approach is also able to reconstruct shape of the object. Another way of using illumination is photometric stereo, where two images for two different light directions are taken from the same view point, and used to recover the surface orientations (Woodham,1978).

<u>Evaluation of Constraints</u>. The aforementioned classification of constraints is useful for finding constraints for specific vision problems. In applying constraints, however, some evaluation criteria are required, and, in most cases, three criteria have to be considered. The first is the cost (time, memory space) of using constraints. In practice, the constraints should be easily constructable and applicable. If too much time is spent in extracting applicable constraints or it takes too much time to propagate constraints, they may be of little practical value, especially when the computational resource is limited.

Another criterion is the robustness (or stability) of the derived constraint. The question to be raised is under how much noise, in the imaging process and the attribute-measuring process, the constraints still hold true. For example, the constraints from point correspondences are less robust than those from line correspondences, since the effect of noise on an image line is averaged out, but that on a point is not. However, constraints from a larger primitive are not always more reliable than those from a smaller primitive. A large primitive detected in the image may include some elements that belong to another primitive, due to the difficulty in separating image regions for different parts of objects. That is, the larger the primitive size, the less reliable the primitive may be. As a result, to have constraints that are insensitive to noise and remain reliable, often physical primitives with identifiable properties (e.g., lines, ellipses, corners, etc.) are used.

The third criterion is the applicable domain of the constraint. It is not easy to find constraints that are independent of application domains. However, constraints that depend too much on its restricted domain may not be desirable, especially when they are to be expanded to a broader domain. For example, the constraints encoded in a junction dictionary (a collection of allowable sets of physically possible line labels for each type of junction) for labelling line drawings may need to be totally revised, when the domain of application is extended to general scenes.

1.2 Approaches to Object Recognition

From the previous discussion, the use of proper constraints is critical to solve a specific problem. Equally important is the way in which constraints are used to solve a particular problem. In this section, the major approaches to the object recognition problem are discussed. The separation of approaches is for the convenience of discussion, and it should be borne in mind that combinations of different approaches are found in the literature on model-based vision.

Research on 3D object recognition can be roughly grouped into five approaches: the invariant property approach, the hypothesis testing approach, the labelling approach, the intermediate representation approach, and the location information approaches.

<u>Invariant Property Approach</u>. This approach usually looks for (mathematical) invariance to unravel the shape, size, depth, orientation, and other properties of objects. The approach has a strong

relation with Gibson's direct approach to biological vision, which holds that perception is a direct response to information presented in the stimulus. Hence, it is proposed that a set of invariances such as texture and motion gradients are picked up to permit the perception of different object properties (Gibson, 1979).

An object property is invariant if it remains the same despite transformation and projection. For example, the ratio of height to width of a rigid object remains invariant as we get nearer to, or farther away from, the object. Since invariances are hard to find for general transformation and projection, invariances are often defined under certain types of transformation (e.g., z-axis-rotation invariant) and projection (e.g., parallel-projection invariant). In some cases, "semi-invariances" (i.e., those which are almost invariant in most situations) are used instead of invariances; e.g., two parallel lines remain almost parallel for most instances of perspective projection.

Finding invariant properties is an important issue in (2D) Pattern Recognition, in which different properties are extracted from the pattern and are fed into a decision mechanism to classify the pattern. The invariant properties are usually global, depending on the whole pattern, and are common to all possible patterns. To classify the pattern, the extracted properties are used to form a feature space. The decision mechanism then uses clustering schemes (surface separation, statistical separation, etc.) to cluster the detected properties in the feature space for a certain type of pattern. The separation schemes are usually parametric functions and their parameters need to be fine-tuned through a training phase.

The success of the application of invariant properties in pattern recognition led to the attempt to use invariances in object recognition. When applied in 2D object recognition, the invariant features can be perimeter of the contour, compactness ($4\pi \cdot$ area / perimeter^2), rectangularity (area of the object contour / area of the its minimum enclosing rectangular), moments of contour (area is the zero-moment; other moments are invariant to object size when divided by the area in the contour). Though these invariances are useful for unoccluded flat objects parallel to the image plane, they are not reliable for arbitrarily oriented 3D objects. One difficulty stems from the fact that, when the object is not flat and can have arbitrary orientation, the invariances do not always exist in 2D image. Another difficulty is that it is hard to segment the image into reliable regions so that the invariant properties can be accurately measured.

<u>Hypothesis Testing Approach</u>. This approach views recognition as the generation of a solution that best fits the sensed cues in the image. In this approach, probable hypotheses about the objects in the image are generated and their validities are tested. In generating hypotheses, it is often desirable that the hypotheses be complete (never misses a solution if one exists), nonredundant (never generates a hypothesis twice), and informed (avoids fruitless hypotheses and favors promising hypotheses). This approach has been very popular in model-based object recognition, and was used by Brooks(1981), Bolles and Horaud (1984), Grimson and Lozano-Pérez (1984), and Pentland (1987).

This approach is closely related to the indirect approach to biological perception, proposed notably by Helmholtz and Ames (Mcburney and Collings, 1983). The indirect approach holds that perception is to make a best guess from the sensed cues about what the object is and where it is. This approach helps to explain several illusions such as Necker's cube, Ames's chair, Ames's trapezoidal window, and the Ames's trapezoidal room.

Labelling Approach. Recognition is regarded as the labelling of the constituents in the image as elements in *a priori* models. Labelling is a correspondence process in which constituents in the image are classified into certain types of primitives (type-labelling), and are matched to the primitives of the same type in a certain model (part-labelling). The object in a region of the image is then recognized as the object whose model primitives best fit the image primitives.

In labelling, the computational complexity tends to increase as the number of primitives to be labelled increases, and as the number of labels each primitive can be given increases. The growth of computational complexity can be exponential if brute-force labelling is used. As an example, consider the problem of labelling each of the M line segments as one of the N generic types; if exhaustive comparisions of each line segment with each label (in this case, line type) is used, M^N or N^M tests will be needed.

To reduce the computational complexity, the constraints of consistent labels are used to prune unnecessary comparisions of primitives and labels. As an example of using constraint satisfaction in type-labelling, consider Waltz's line labelling algorithm. In Waltz's formulation, a line segment in the image can be labelled as three types: concave edge, convex edge, or occluding edge. To reduce the labelling effort, a junction dictionary containing physically possible interpretations of different vertices is used; e.g., for an arrow vertex of three lines, there are three physically possible combinations of concave, convex, and occluding line labels. To use the dictionary of constraints, Waltz developed an filtering algorithm to efficiently root out impossible labels. The algorithm is duplicated from Davis(1987) as follows:

```
procedure REFINE(C,$X_i$){
   /* REFINE updates allowable labels for a specific argument $X_i$
      in a given constraint C, and returns the updated set of
      allowable labels */
   /* C is a constraint over arguments: $X_1$, $X_2$, $X_3$, ....., $X_k$;
      e.g., in Figure 1.2.(a), parallel(a,b) has two arguments:
      primitive "a" and "b". */
   /* The set $S_i$ contains allowable labels for $X_i$; e.g., the image
      primitive "b" in Figure 1.2.(a) is a line, and it can be
      label as model line "2" or "3". Thus, the allowable set for
      image primitive "b" is {"2","3"}. With the constraint
      "intersect(b,c)", the allowable set for primitive "b" can
      be reduced to {"3"}.
   /* $A_i$ is an atom (or label) in the allowable label set $S_i$ */

   S = $\emptyset$;
   for each $A_i \in S_i$ do{
       if there exist a set
          {($A_1$,....$A_k$)| $A_j \in S_j$ for j=1,...,k , $j \neq i$}
          such that C($A_1$,.....,$A_i$,.....$A_k$) holds
       then S $\leftarrow$ S $\cup$ {$A_i$};
   }
   return S;
}

procedure REVISE(C($X_1$,....,$X_k$)){
   /* The set $S_i$ contains allowable labels for $X_i$ before
      the procedure REVISE is called */
   /* The procedure updates the allowable labels for all arguments
      $X_1$,.....,$X_k$ of a given constraint C, and return the set of
      arguments whose allowable lable sets were changed */

   CHANGED $\leftarrow \emptyset$;
   for each argument $X_i$ do{
      S $\leftarrow$ REFINE(C,$X_i$)
      if S = $\emptyset$ then halt;/* the constraints were inconsistent */
      else if S $\neq S_i$ then{
            $S_i$ $\leftarrow$ S;
            add $X_i$ to CHANGED;
      }
   }
   return CHANGED;
}
```

```
procedure WALTZ(Q){
    /* Q is a queue of constraints over arguments $X_1,.....X_k$ */
    while Q $\neq \emptyset$ do{
        remove a constraint C from Q;
        CHANGED $\leftarrow$ REVISE(C);
        for each $X_i$ in CHANGED do{
            for each C' $\neq$ C which has $X_i$ in its domain do
                add C' to Q;
        }
    }
}
```

The computational complexity of the Waltz algorithm is shown (Mackworth and Freuder, 1985) to be O(ae), where a is the number of possible labels per argument (or node), and e is the the number of constraints. Although the Waltz algorithm is very efficient for constraints of order relations, it tends to go into infinite loops for constraints of linear or nonlinear algebraic relations (Davis, 1987).

Another widely used labelling techinique is the parallel relaxation labelling, which was originally developed for assigning numeric or symbolic labels to objects in the presence of ambiguity. In relaxation labelling, constraints usually are expressed as compatibility measures

$$C_{ij}(A_i,A_j),\ \text{for } A_i \in S_i,\ A_j \in S_j,\ 1 \le i,j \le k.$$

Two approaches can be taken for the compatibility measures: one is discrete relaxation where pairs of labels are either compatible or completely incompatible, and the other is the continuous relaxation where compatibility measures are real numbers with each magnitude for the level of compatibility and sign for positive consistency or negative consistency (or inconsistency). One example of the relaxation labelling technique is given by Hummel and Zucker (1983).

Intermediate representation approach. This approach takes the position that a general purpose vision system is possible only if certain information is made explicit. The information can be "$2\frac{1}{2}$D" sketch (Marr, 1983) and intrinsic images (Tenenbaum, Barrow, and Bolles, 1979). A $2\frac{1}{2}$D sketch is the viewer-centered representation of depths, local surface orientations, and surface discontinuities (obtained from such cues as stereopsis, structure from motion, optical flow, occluding contour, surface orientation contours, texture, and shading). The intrisic images are again viewer-centered maps of surface distances, surface orientations, surface reflectances, illuminations, and surface textures.

Realizing that direct 3D object recognition from intensity images is difficult, most researchers have tried to use intermediate image representations such as range images for object recognition. The range image can be obtained directly from a ultrasonic or laser range finder or the result of shape-from-X techniques operating on intensity images. Examples of using range data or local surface orientations for object recognition are Oshima and Shirai (1983), Grimson and Lozano-Pérez (1984,1985), Bolles and Horaud (1985), Faugeras and Hebert (1986), and Pentland (1987). However, these methods with range images are still quite complex and time-comsuming in both data aquisition and interpretation.

Location-information approach. This approach holds that using the (estimated) object location information saves time in interpreting images. The role of location information can be seen from the example of recognizing an object whose location in space is known (e.g., recognizing a workpiece held by a fixturing device). Obviously, using the known object location with other property/relation constraints to match image primitives and model primitives is more efficient than using property/relation constraints alone. As a second example, if the object is known to be flat, it will be simpler to perform matching by using one rotation parameter and two translation parameters to represent the object location instead of using all six parameters. The technique is again to use the known information about object location.

What happens if the object location is not restricted and no *a priori* location information is available? There are three obvious ways one can take to match primitives between an image and a model. The first is to apply only labelling techniques by using part properties and relations. The object location is then recovered after all consistent matches are obtained. Clearly, this is the pure labelling approach, as discussed above.

An alternative way is to implant the location parameters into the constraints with the constraints being used in two directions. In one direction, constraints are propagated over unknown parameters to update parameter intervals. If the updated interval is empty, then the set of constraints is unsatisfiable and thus inconsistent. In the other direction, the updated intervals of unknown parameters are used to calculate the values of expressions for properties or relations of other primitives. The ranges of the expressions are used to predict the appearances of the associated primitives and to reduce the search space for forming a plausible interpretation. One such example of using constraints for testing consistency and predicting possible matches between object and image primitives is ACRONYM by Brooks (1981). Another example by the author is to be introduced in chapter 2;

however, experiments showed that it is not as efficient as the third way of using location information, to be discussed next.

The third way is to initially hypothesize a small set of matches (e.g, three line correspondences or one elliptical correspondence) between the image primitives and the model primitives by using properties and relations of primitives. This step is then followed by estimating the object location with the hypothesized correspondences. The recovered location is used to predict attributes of image primitives (e.g., visibility, the slope or Y-intercept of a line, and center and range of an ellipse in the image), and the prediction is used to augment the set of hypothesized matches of object primitives to image primitives. The hypothesized matches are then used to refine the estimation of object location, and are tested against the recovered location to see if they are consistent. If all hypothesized matches are tested to be consistent with the estimated location, the hypothesized matches are consistent, and the cycle of prediction, augmentation, localization, and consistency testing continues. If, however, the hypothesized matches are tested to be inconsistent with the recovered location, the current set of hypothesized matches is rejected, and other sets of hypothesized matches are used. Because the use of the recovered object location enables the rejection of inconsistent matches at an early stage, this approach augments the set of consistent matches without trying many fruitless paths and with minimum backtracking. Examples of this approach are SCERPO by Lowe (1987), the alignment approach by Huttenlocher and Ullman (1987), and the proposed locating-labeling approach (in Chapter 2).

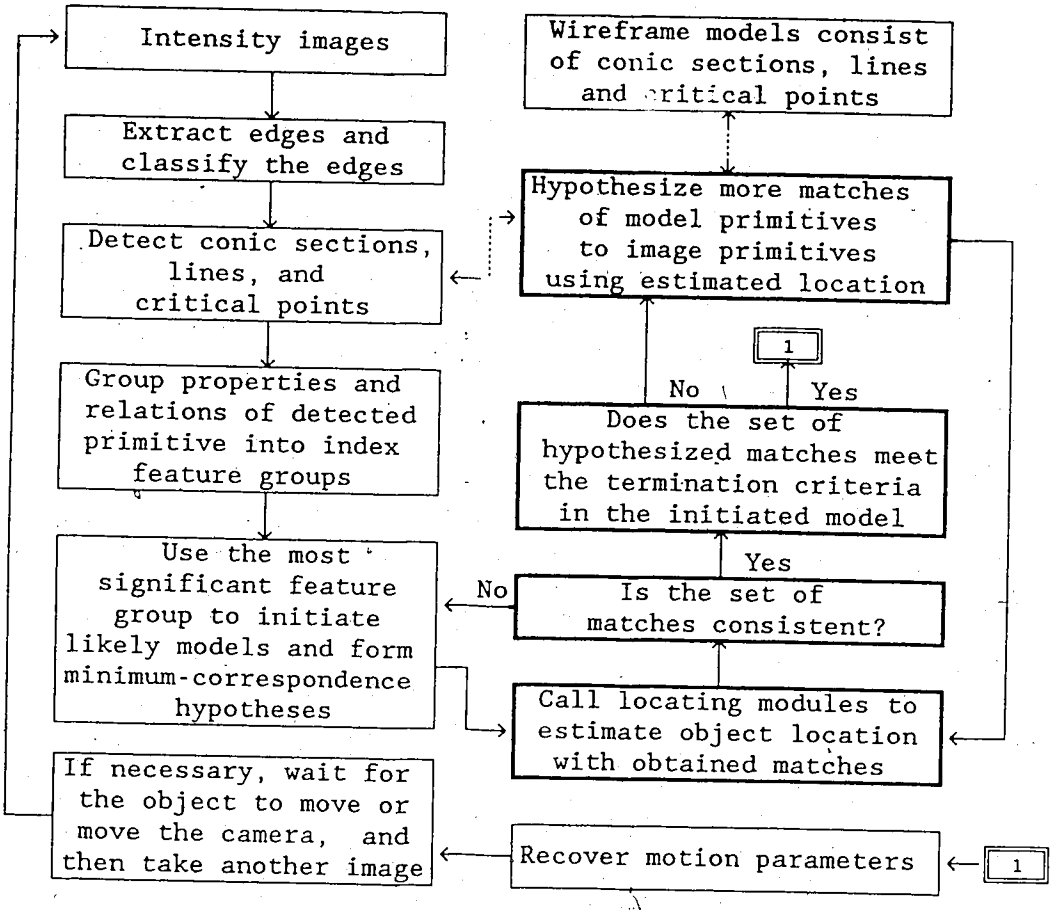
1.3 The Approach Used In This Thesis

For object recognition, the approaches outlined in the previous section can be used concurrently in a vision system. For instance, invariant properties (in intrinsic images, $2\frac{1}{2}$D sketch, or intensity images) can be used to hypothesize the model, and the test of hypothesis can be done with constraint satisfaction. Thus, the approach used in this dissertation for object recognition is not a single one of the five approaches given above, but a combination of all five. However, it is the purpose of this thesis to stress the importance of index feature grouping and location information in object recognition. Hence, a major part of this thesis will emphasize the use of index feature grouping (IFG) and location information in recognizing objects. For distinction, the proposed approach to object recognition is called the "locating-labelling" method because the correspondence process uses recovered location information to guide the matching of model primitives to image primitives. In order to locate an object in 3D space from correspondences between conic primitives in the model and a given image, analytic solutions using various types and number of correspondences of conic primitives are derived (Conics are commonly found in man-made objects and can be easily extracted from an intensity image). In addition to recognition and localization of static objects, this thesis also discusses the recovery of object motion relative to the camera. The recovered motion is useful for tracking moving objects and can provide verification information for the object recognition process.

The images used here are intensity images taken by a video camera. The results show that "absolute" 3D orientations and positions of objects can be recovered by using a static image. Further, 3D motion parameters (three for rotation and three for translation) between two frames can be recovered accurately. Above all, this thesis shows that, with proper use of location information, 3D object recognition is possible and economical by working on an intensity image alone.

The work in this thesis is briefly summarized in the block diagram in Figure 1.4 and is discussed in the following chapters. In Chapter 2, the problem of object recognition is addressed. The representation of object models, the detection of conic primitives in an image, the grouping of index features, the formulation of hypothesized matches, and the test and augmentation of hypothesized matches are discussed.

In Chapter 3, the problem of object localization from an intensity image is addressed. An analytic treatment of locating an object from conic correspondences is given. It will be shown that the minimum number of correspondences to locate an object is three for line or point correspondence and one for ellipse correpondence.



Intensity images
Extract edges and classify the edges
Detect conic sections, lines, and critical points
Group properties and relations of detected primitive into index feature groups
Use the most significant feature group to initiate likely models and form minimum-correspondence hypotheses
If necessary, wait for the object to move or move the camera, and then take another image
Wireframe models consist of conic sections, lines and critical points
Hypothesize more matches of model primitives to image primitives using estimated location
No
Yes
1
Does the set of hypothesized matches meet the termination criteria in the initiated model
Yes
No
Is the set of matches consistent?
Call locating modules to estimate object location with obtained matches
Recover motion parameters
1

Figure 1.4 Block diagram of the proposed method. The solid lines are for control flow, and dashed lines are for data flow. The bold line boxes are for the iterative processes of hypothesis testing and augmentation. For multiple images, initiation of likely models and the formation of minimum number of matches can use the result of previous image; e.g., the object classes and object locations.

In Chapter 4, the problem of motion recovery from a sequence of intensity images is solved with an extended Kalman filter (EKF) approach. In the EKF formulation, the dynamic model of object motion and the measurement model of image primitive attributes are constructed and the motion parameters in the dynamic model are estimated such that an uncertainty measure is minimized. The primitives used in recovering motion are again conics.

The algorithms derived in Chapters 2, 3, and 4 are implemented and discussed in Chapter 5. Three separate experiments are given. The first is the camera calbration to determine the focal length of the camera with its focusing lens at a certain position. The second is a partial implementation of the "locating-labelling paradigm" to 3D object recognition. The third is an implementation of motion recovery using only point correspondences. Finally, Chapter 6 contains conclusions.

Throughout this thesis, perspective projection is used to relate image coordinates (X,Y) of an image point to 3D coordinates (x^c, y^c, z^c), relative to the camera, of the corresponding point in space. The relationship is given by:

$$X = -f\, x^c / z^c \qquad (1.1.a)$$

$$Y = -f\, y^c / z^c \qquad (1.1.b)$$

where f is the focal length of the camera with focusing lens at a certain position.

In characterizing an object's location, we use an orientation matrix O and a position vector **p** to transform the coordinates of points in a selected object coordinate basis to the coordinates in the camera coordinate basis. The transformation is given by:

$$\begin{bmatrix}x^c\\y^c\\z^c\end{bmatrix} = O \begin{bmatrix}x^o\\y^o\\z^o\end{bmatrix} + p \tag{1.2}$$

where x^o, y^o, and z^o are the coordinates of a point in the object coordinate basis, and x^c, y^c, and z^c are the coordinates of the same point in the camera coordinate basis

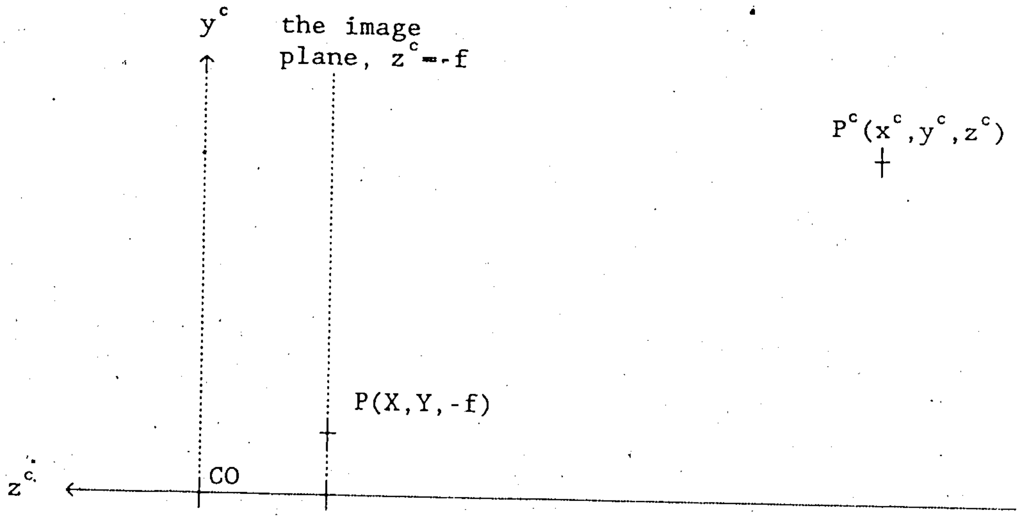


Figure 1.5 The perspective projection of a point P^c in space. The projected image point is the intersection of the image plane $z^c=-f$ and the line through the camera origin CO and the point P^c. Thus, $X=-fx^c/z^c$ and $Y=-fy^c/z^c$ define the image point P in the image plane.

Chapter 2. Object Recognition

Model-based object recognition is the determination of an object model which best projects a collection of detected primitives and their relations in the image. Due to occlusion or overlapping, detected primitives belonging to an object may be separated by primitives from the same or other objects. As a result, some primitives and relations of a visible object may not appear in the image. Consequently, a set of disjointed primitives and incomplete relations is recognized as a projection from a certain object only if a set of consistent correspondences between the detected features (here, features are used for both properties and relations of primitives) and features of a given model exists. Note that this is a necessary condition for recognition, because different objects can produce the same set of image primitives and relations.

In finding the best-fit model, three components are needed in the proposed recognition system. These are the representation of the object models, the extraction of image primitives and key feature groups, and the matching between the image primitives and model primitives. The three components are usually interrelated, thus the design of one component usually affects the other two. For example, the representation of a model affects the type of primitives and relations which can be extracted from the image and the way of the matching which can be carried out. Despite this interrelationship, these three components will be discussed separately in the following sections, with some overlapping for the sake of clarity. The discussion will be confined to the condition that only intensity images are used, and no intermediate representation such as $2\frac{1}{2}$D sketches or intrinsic images is used.

Prior to a detailed discussion, each component is briefly summarized here. The object models are currently wireframes of primitives such as conic arcs, lines, and points (critical points for maxima of boundary curvature or vertices for edge junctions). Also, numeric and symbolic knowledge about the properties and relations of primitives in a object are encoded in the model for that object. The encoding is done by using nodes and links from semantic nets and rules from production systems. If necessary, the representation of models can be expanded in primitives (e.g., the use of surfaces) or in properties and relations (e.g., the use of shading, colour, and texture).

Corresponding to models, the image primitives to be extracted are conics, lines, and critical points. The extraction of primitives is done by a subdivision algorithm which recursively breaks boundary curves (or detected edges in the image) into line segments, and then by a conic fitter that combines consecutive line segments into conic arcs. Once primitives are extracted, neighboring primitives are grouped together by using their properties and relations to form feature groups for initiating likely object models from which the detected image primitives originate. Examples of feature groups are an ellipse, two nearly parallel lines, a vertex of three lines, etc.

The matching of primitives between images and models can be done in two directions: bottom-up (data-driven) and top-down (model-driven). In either direction, constraints from properties and relations of primitives and the location information are used to reduce the search effort. At an early stage, the matching takes the data-driven direction by using the most significant feature group through an index table to retrieve likely object models. The correspondences of primitives between each initiated object model and the selected feature group are used to form hypotheses. Each formulated hypothesis is tested by a hypothesis-tester using the object location recovered from the obtained correspondences. In testing a hypothesis, the hypothesis-tester predicts additional correspondences and augments the set of correspondences until an inconsistency is detected or a confirmation criterion in the initiated model is satisfied by the set of correspondences.

2.1 Object Models and Database

Each *a priori* object model used here consists of three kinds of elements: nodes for shape primitives, links for relations among primitives, and rules for matching actions. The nodes and links are from the notations of semantic network and graph theory, and rules are similar to production rules. A node represents a shape primitive and its properties. A primitive can be a critical point, a line, a conic section, or a surface (currently, surfaces are only used for predicting visibilities of other primitives and not directly used in matching). Each node contains a property list, which consists of geometrical properties and photometric properties. Examples of geometrical properties are: primitive type, geometrical attributes of the primitive, and relative coordinates to a reference point in the object model. Examples of photometric properties are colour, reflectance, and texture. The photometric properties are not used at present because surfaces are not used directly for matching.

Links tie multiple nodes together to represent relations between the primitives at the nodes. A relation usually has at least two links for different nodes. Possible symbolic relations are parallelism, intersection, connectedness, proximity, inclusion, coplanarity, collinearity, symmetry. Numeric relations are the coordinate transformations between two primitives. A coordinate transformation between two primitives is not explicitly expressed, but is implicitly encoded in the relative coordinates of the primitive to a reference coordinate basis in the model.

A rule usually contains abstracted knowledge about the object model, and is used to guide the matching process. Three kinds of rules are used at different stages of matching. At first, after likely models are initiated by the index table (to be explained later in Section 2.3) as candidates, rules in each model are used to produce a set of correspondences between the image features and model features and to generate a confidence measure for each model as a candidate. The confidence measure is then used to schedule the likely models. Secondly, when a likely model is selected, a rule in the model would tell the matcher what information to look for and where to look for it in order to formulate a set of minimum correspondences sufficient for testing the candidate model (the term "minimum correspondences" is for the minimum number of matches needed to recover an object's location, see Chapter 3 for details). As an example, when two parallel lines match two lines in the image, a rule would ask the matcher to select a neighboring line, not parallel to the matched two lines in the model, as the model line to be paired to a line in the image. Naturally, the probable image lines would also be close to the two matched image lines. The set of minumum correspondences is used to form a hypothesis and then passed to the hypothesis-tester. The hypothesis-tester augments the correspondences between the image and the initiated model to test the plausibility of a hypothesis. The third type of rule is used by the hypothesis-tester, with the recovered object location, to predict visibilities of other unmatched primitives and possible area of the primitives, if visible, in the image. The probability of a primitive's existence in the image and the reliability of measuring the primitive's attributes (for example, a longer line has more reliable attributes, when measured, than a shorter line) are used to rank the primitives for the hypothesis-tester to use.

In addition to the object models, the database of the proposed system contains an index table to initiate likely models by using detected image feature groups, as is illustrated in Figure 2.1. The index table is generated by using the object models and has image feature groups in its entry column. For each entry, the second column contains pointers to the models which can produce the image feature group through perspective projection. An image feature group is a combination of image primitives and their relations. For example, an ellipse forms an image feature group which contains only one primitive, whereas two parallel lines form an image feature group which contain two primitives and one relation.

Input: index feature groups ⇓	Output: likely models ⇑
ellipse(P_1)	M_1,M_2,M_3, ——
intersection(P_1,P_2) $\wedge$ellipse(P_1)$\wedge$line(P_2)	M_1,M_3, ——
near-parallel(P_1,P_2) $\wedge$line(P_1)$\wedge$line(P_2)	M_2,M_4,M_5, ——
near-perpendicular(P_1,P_2) $\wedge$line(P_1)$\wedge$line(P_2)	M_2,M_5, ——
intersection(P_1,P_2,P_3) $\wedge$line(P_1)$\wedge$line(P_2)$\wedge$line(P_3)	M_1,M_2,M_3,M_4,M_5, ——
trapezoid(P_1,P ,P_3,P_4)$\wedge$line(P_1) $\wedge$line(P_2)$\wedge$line(P_3)$\wedge$line(P_4)	M_6,M_7, ——
U-shape(P_1,P_2,P_3) $\wedge$line(P_1)$\wedge$line(P_2)$\wedge$line(P_3)	M_6,M_7,M_8, ——
consecutive(P_1,P_2,P_3) $\wedge$line(P_1)$\wedge$ellipse(P_2)$\wedge$line(P_3)	M_9, ——
—— —— ——	—— —— ——

Figure 2.1 An example of index table for initiating likely object models. The input pattern is the selected image feature group which is expressed in conjunction normal form. The output is a list of models which are likely to produce the selected feature group in the image. In the table, P_i is the i-th detected image primitive, and M_j is the j-th model in database.

2.2 Primitive Detection and Feature Grouping

Edge Extraction. Edges (image pixels at which image intensity changes) are detected from a given intensity image, by convolving the intensity image with the Laplacian of a Gaussian low-pass filter or $\nabla^2 G$, where G is the Gaussian low-pass filter. The zero-crossings of the convolved image then give edges one pixel in width. Alternatively, edges can be detected by convolving the image with other difference operators such as the Sobel operator to give the magnitude of the intensity gradient at each pixel. The edges in the convolved image are thinned to single pixel widths by picking up the pixel with maximum gradient magnitude out of pixels across the edge. Due to the fact that zero-crossings of the image convolved with $\nabla^2 G$ may not always correspond to significant intensity changes in the image, especially where the intensity gradient is low (Lowe, 1987), the Sobel operator approach is chosen in this implementation. The choice here is primarily for the convenience of the current implementation since our main interest is not focused upon optimal edge extraction techniques.

Edge Classification. An edge detected in the image can be an occluding boundary (arising from discontinuity in depth to the view point), a surface boundary (arising from surface orientation change, albedo change, a texture boundary, or just a curve on the surface), or a shadow edge (arising from illumination discontinuity). Consequently, it is advantageous to classify edges into one of these types before enacting a shape matching process to extract conic primitives. One way to classify an edge as a shadow boundary, an occluding boundary, or a surface boundary is proposed by Tenenbaum, Barrow, and Bolles (1978), using the intensity changes on both sides of the edge. Another way to distinguish convex from concave edges by using the intensity profile across an edge is proposed by Horn (1977). He shows that the convex edge usually has step changes in the intensity profile, while a concave edge usully has roof-shaped changes.

In the current implementation, however, edges are not classified, since this thesis is focused on the competence of the proposed "locating-labeling method" for object recognition, not the details of all possible representations of information. Therefore, the thinned edge image is fed directly into the primitive detector for detecting critical points, straight lines, and conics in the image.

Primitive Detection. Initial image primitive identification is implemented in a two-stage algorithm. In the first stage, each boundary curve (or edge) is divided into straight line segments by breaking the curve at points of local maxima of curvature. As a result, each boundary curve is approximated by a polyline with a list of break points. The second stage fits conic sections to subsets of the polyline-approximated curve. On completing this step, a list of conic arcs is generated for the given boundary curve. Also, the points which separate two consecutive conic arcs longer than a prescribed length are selected as critical points. This two-stage algorithm is similar to that of Liao (1981) with minor modifications.

The approximation of a boundary curve in the image with a polyline is done by using a recursive subdivision algorithm. The algorithm breaks a given curve at the point most distant from the straight line which passes through the two endpoints of the given curve. Each newly obtained subcurve is tested to see either if it can be fit by a straight line with distances of points on the subcurve to the approximating straight line below a prescribed limit or if the length of the subcurve has fallen below a prescribed minimum length (e.g., 6 pixels). The subcurve is accepted as one segment of the polyline when one of the above criteria is satisfied; or else, the subdivision algorithm breaks the subcurve recursively. This process is summarized in the procedure SPLIT() given below.

```
Procedure FAREST(E){
   /* The edge E contains n points $(X_1,Y_1)$, $(X_2,Y_2)$, ...., $(X_n,Y_n)$ */
   /* the routine ALLOWABLE_DEVIATION() is a function calculating the
      allowable fitting error according to the edge's length */
   $m \leftarrow (Y_n-Y_1)/(X_n-X_1)$ and $T \leftarrow Y_1 - m X_1$;
   $i \leftarrow 1$ and $d_{max} \leftarrow 0.$;
   while $i \le n$ do {
        $d \leftarrow \|Y_i - m X_i - T\|$;
        if $d > d_{max}$ then $j \leftarrow i$ and $d_{max} \leftarrow d$;
        $i \leftarrow i + 1$;
   }
   $\epsilon \leftarrow$ ALLOWABLE_DEVIATION(n);

   if $d_{max} / (1+m^2)^{1/2} < \epsilon$ then return 0;
   else return j;
}
procedure SPLIT(E){
   /* The edge E contains n points $(X_1,Y_1)$, $(X_2,Y_2)$, ...., $(X_n,Y_n)$ */
   $S \leftarrow \emptyset$;
   if n [illegible] minimum_length then return {P};
   $j \leftarrow$ FAREST(E);
   if j = 0 return {P};
   $E_1 \leftarrow \{(X_1,Y_1), (X_2,Y_2), ...., (X_j,Y_j)\}$;
   $E_2 \leftarrow \{(X_{j+1},Y_{j+1}), (X_{j+2},Y_{j+2}), ..... (X_n,Y_n)\}$;
   $S \leftarrow S \cup$ SPLIT($E_1$);
   $S \leftarrow S \cup$ SPLIT($E_2$);
   return S;
}
```

When the subdivision algorithm terminates, we have a list of line segments and a list of breakpoints for the boundary curve in the image. However, these breakpoints may not correspond to points with local maxima of curvature. One reason is that the number of breakpoints depends on the resolution of the prescribed error limit and the minimum length. As can be seen, if the error limit of fitting line segments to a curve decreases, then the number of breakpoints, in general, increases. Another reason is that the breakpoint from the first subdivision is not necessarily a high curvature point when the boundary curve is closed (for this reason, the breakpoint of the first subdivision can be chosen at the midpoint of the boundary curve). To refine the breakpoints for the boundary curve, a readjustment procedure is needed. The readjustment is done by successively testing if two consecutive segments of the polyline-approximated curve can be merged into a single segment to fit a straight line within the prescribed error limit of distance. If so, the new segment replaces the two consecutive segments. If not, a point on either of the two consecutive segments which breaks the combination of the two consecutive segments into another two segments with smaller fitting error than before is selected. The two new segments are used to replace the two previous ones. This refinement is repeated until no further readjustment of breakpoints can be made.

After the polyline approximation of a boundary curve, the conic sections are fit to subsets of successive segments of the boundary curve to have the curve approximated with a list of conic segments. This is done by successively testing if two consecutive segments can be fit to a conic equation

$$a_1X_i^2 + a_2X_iY_i + a_3Y_i^2 + a_4X_i + a_5Y_i + 1 = 0$$

where the pair (X_i, Y_i) represents every point in the two segments.

By using the pseudo-inverse method to minimize the squared sum of the equation errors

$$f(X_i,Y_i) = \hat{a}_1X_i^2 + \hat{a}_2X_iY_i + \hat{a}_3Y_i^2 + \hat{a}_4X_i + \hat{a}_5Y_i + 1$$

for all points in the two segments, we can obtain the estimated conic parameters: $\hat{a}_1$, $\hat{a}_2$, $\hat{a}_3$, $\hat{a}_4$, and $\hat{a}_5$. The fit is acceptable if an error measure is within a required limit. The error measure is defined as:

$$E = \mathrm{Max}\{\mathrm{Min}[d_x(X_j,Y_j),d_y(X_j,Y_j)] \mid \text{for all breakpoints, } j\text{, in } C\}$$

where C is the union of the two consecutive segments. The first segment in C can have multiple breakpoints, as it can be a previously merged conic segment. The measure $d_x(X_j,Y_j)$ is the distance of the point (X_j,Y_j) from the intersection of the calculated conic arc and the line $Y = Y_j$. Likewise, $d_y(X_j,Y_j)$ is the distance of the point (X_j,Y_j) from the intersection of the calculated conic arc and the line $X = X_j$.

If the fitting of conic equation is acceptable, the two consecutive segments are assimilated into a new conic segment. The fitting then continues for the new conic segment and the next line segment in the boundary curve. If, however, the fitting is not acceptable, the first of the two consecutive segments is refitted by using a nonlinear conic fitter (to be discussed later in this section) and then saved as such, and a new fitting process begins for the second segment and the next line segment in the boundary curve. This merging process terminates when all line segments of the curve are tried.

The aforementioned conic fitter (which is linear in the conic parameters) minimized the squared sum of equation errors, which may not always correspond to the minimization of conic parameter error defined as $\sum_{k=1}^{5}(a_k - \hat{a}_k)^2$, where a_k is the actual paramter usually not accessible and $\hat{a}_k$ is the estimated one. In fact, the linear conic fitter favors curves with high average radius of curvature value (Ballard and Brown, 1982). Thus, a hyperbolic equation may result when attempting to fit a partial elliptic curve in the image (especially when the partial ellipse is smaller than half of the entire ellipse). To rectify this situation, each conic arc, prior to being saved, is refined by minimizing the error (Ballard and Brown, 1982):

$$e^2 = \sum_{i=1}^{n} \left[\frac{f(X_i,Y_i)}{\|\nabla f(X_i,Y_i)\|} \right]^2$$

where $f(X_i,Y_i)$ is the equation error as defined above, and ∇ is the gradient operator. The pair, (X_i,Y_i), is for every point in the conic arc, and n is the number of points in the arc.

After the two-stage algorithm terminates, there will be a list of conic arcs and of breakpoints. The breakpoints are processed further to generate stable critical points. Stable critical points are those breakpoints whose two neighboring segments are longer than a required length (e.g., 50 pixels). This process is to eliminate the breakpoints in a noisy portion of a curve (Fischler and Bolles, 1986).

For the detected primitives, the significance and reliability are determined. Significance is a strength measure for the evidence of the presence of a certain object, and reliability is an accuracy indicator of the measured attributes or parameters. In general, a full ellipse is most significant and most reliable. Partial conics can be significant but their computed parameters may not be reliable. Straight lines are less significant than conics of nonzero curvature, but the measured attributes of straight lines are more reliable than those of partial conics. Obviously, longer lines have more reliable computed attributes and more significance than shorter lines. Overall, critical points have the least reliable attributes, except for junctions of two or more edges. For comparision of significance and reliability of conic primitives, see Table 2.1 and Table 2.2.

Table 2.1 Reliability of Measured Attributes for Conic Primitives (in decending order)

1. Full Ellipse
2. Ellipse or hyperbola with spaning angle > π
3. long line (e.g., > 20 pixels)
4. ellipse or hyperbola with spaning angle < π
5. short line (e.g., > 6 pixels)
6. vertex of edges
7. critical point

Note. Reliability is an accuracy indicator of the measured attributes or calculated parameters.

Table 2.2 Significance of Detected Conic Primitives (in decending order)

1. Ellipse or hyperbola (the larger spaning angle the more significant)
2. line (the longer the more significant)
3. vertex of edges
4. critical point

Note. Significance is a strength measure for the evidence of the presence of a certain object. For individual image primitives, the significance is listed in this table. However, after index feature grouping, the significance may change; e.g., a four-line quadrilateral may be more significant than a single ellipse in indexing the originating object. For index feature groups, see Figure 2.1.

Representation of Detected Primitives. After primitives are detected, they are represented and arranged in order that relations between primitives can be easily obtained. For example, parallel lines are those lines with similar slope value. Another reason for representing detected image primitives is that, when model-driven matching is used, image primitives which are probable candidates for a given model primitive can be easily retrieved. In the current implementation, three types of cells (data structure) are used separately for the three types of primitives: ellipse and hyperbola, line, and point, as shown in Figure 2.2.

When an ellipse (or hyperbola) is detected, its two endpoints, arc length, parameters, enclosing rectangle, and spanning angle are captured in a cell structure, like the one in Figure 2.2.(a). The arc length is defined as the number of pixels in the detected arc. The parameters are the conic coefficients calculated by the conic fitter. The enclosing rectangle is defined as the smallest rectangle which encompasses the detected arc in the image. The rectangle is expressed by two points. One is the upper-left corner, the other is the lower-right corner of the rectangle. These two corners are expressed in image coordinates, i.e., the set of 512 pixels in X-axis and 480 pixels in Y-axis. The spanning angle is defined as the angle between two diameters (lines through the center of ellipse or hyperbola) passing through the endpoints of the detected arc. For example, a full ellipse has a spanning angle of 2π, and a half ellipse has a spanning angle of π.

The elliptic or hyperbolic cells are linked to create a table by using the Y-coordinate of the upper-left corner of the enclosing rectangle as its column ordinal, and the X-coordinate as the row ordinal (as in Figure 2.3). In addition to the cell table, a cell list containing pointers to conic cells is created. The list is sorted by using the arc length as ordinals so that the cell with longer arc are placed ahead of those with shorter arcs.

When a line is detected, its two endpoints, length, slope, and Y-intercept are stored in the cell structure, as shown in Figure 2.2.(b). The slope, m, and Y-intercept, T, are obtained by fitting the equation, $Y = m X + T$, to all points in the detected line. In the case where $m \simeq \infty$ or $\|m\| > 500.$, $1/m$ and X-intercept are stored instead of m and Y-intercept. Like conic primitives, a table in terms of two endpoints of each line is created. The table is similar to the one in Figure 2.3, except that both endpoints of the line are used as entry keys. That is, each line cell is pointed to by the table twice instead of once. In addition to the table, two lists of pointers to line cells is created. One is sorted by the line length, and the other by the line slope.

Point cells like Figure 2.2.(c) are used for the detected points. Like conic and line cells, a table in terms of the point coordinates are created for all point cells. The table is similar to the one in Figure 2.3.

conic cell
type : ellipse or hypebola length: number of pixels endpoints: $P_1(X_1,Y_1),P_2(X_2,Y_2)$ parameters: a_1,a_2,a_3,a_4,a_5 center: $P_c(X_c,Y_c)$ area: $P_{ul}(X_{ul},Y_{ul}),P_{lr}(X_{lr},Y_{lr})$ angle: between P_1P_c and P_2P_c

(a)

line cell
length: number of pixels endpoints: $P_1(X_1,Y_1),P_2(X_2,Y_2)$ slope: m intercept: T

(b)

point cell
coordinate: P(X,Y) edges: conic or line cells

(c)

Figure 2.2 Three kinds of cells for three types of primitives.

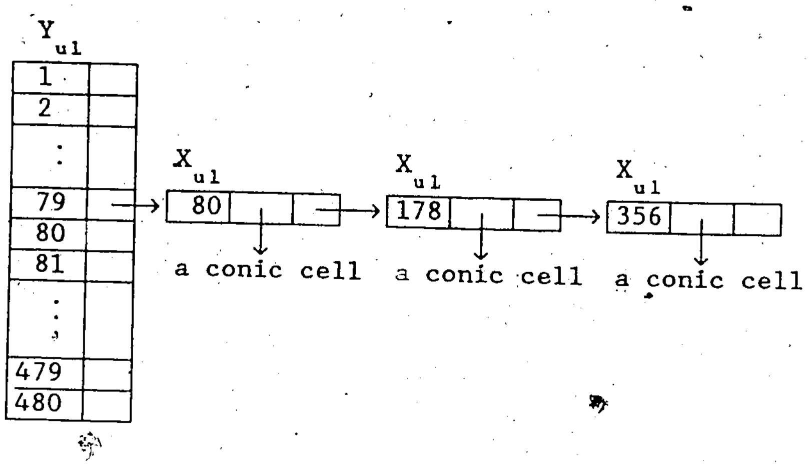


Figure 2.3 A conic cell table using the upper-left corner of the conic cell's enclosing rectangle as the sorting key. The column contains two fields; the first contains the value of Y_{ul}, which is used as the entry key, and the second is a pointer to a row of items with the same value of Y_{ul}. Items in the same row are sorted according to X_{ul} value. Each item in rows has three fields. The first field is for the value of X_{ul}, the second contains a pointer to the corresponding conic cell, and the third contains a pointer to the next item in the row which has larger value in X_{ul} than that of the current item.

Index Feature Grouping. Although the primitives detected by using the aforementioned method can be matched directly to models, it is still uncertain where in the image the matching process should start and which object model should be selected prior to others. One possible solution is to use feature groups of primitive properties and relations. A feature group is a combination of the properties and relations of nearby primitives. After features are grouped, the most significant feature group is used to initiate likely models and to formulate hypotheses for matching. Simply put, the index feature grouping is to detect the most significant cluster of primitives and their relations in the image, and it is used to formulate probable hypotheses of correspondences between model primitives and image primitives.

Index feature groups can include a full conic section, conic arcs or lines which intersect or neighbor each other, parallel lines, perpendicular lines, U shape of three lines, trapezoid shape of four lines (Lowe, 1986), skewed symmetry of four lines (Kanade, 1981), vertex of multiple lines or curves (Guzman, 1968 and Waltz, 1975), number of critical points in a boundary curve, composition of conic and line segment of a boundary curve, or Fourier coefficients of a boundary curve. Currently, only intersecting conic arcs or lines, parallel or perpendicular lines, and vertices are used. Once index feature groups are obtained, each group is expressed in a conjunctive normal form (like those in the left column of Figure 2.1), and is used to match the preconditions of rules in the entry columns of the index table. Figure 2.1 shows an example of an index table with the precondition in its left column.

It should be noted that an index feature group is used to index likely models, and it does not necessarily have a model counterpart (i.e., no entry for it in the index table), since the group may be too big and may include parts from different objects. Hence, there is a trade-off between the size of the group and the reliability of the group. The reliability of the group diminishes as the size or the area of a feature group grows. This is generally true when one attempts to cluster a group of image elements and then given the group an abstracted name for later processing, as is often used in icon-based computational vision.

The index feature grouping is different from image segmentation. Segmentation or parsing, in general, tries to separate image regions corresponding to different objects so that the recognition can be done by finding which object can generate the characteristic in a certain region. Despite a lot of literature on segmentation, it is generally agreed that segmentation of intensity image is very difficult, as different objects may produce the same visual characteristics (e.g., intensity, shading, texture, etc.), and object boundaries may be obscured by shadows or occlusions. In contrast, the index feature grouping is used to form hypotheses about the detected features without forcefully segmenting the image region. The hypotheses are subsequently tested to be rejected or augmented by a hypotheses tester (to be discussed in next section).

The index feature grouping (IFG) is similar to the local feature focus (LFF) by Bolles and Cain (1982). The IFG and LFF both start with the significant features and concentrate on their neighborhoods, but one major difference between the two approaches is in the matching stage. The cluster of features generated by LFF is matched to object models by using a graph-matching technique to establish the largest set of correspondences between the cluster of image features and features of a selected model. The graph-matching used was a maximal-clique algorithm. A clique of size n is a graph of size n whose nodes are totally connected to each other with arcs. The maximal-clique algorithm determines a completely connected subgraph of maximum size from a given graph. The nodes in Bolles and Cain's graph are the possible correspondences between image features and model features. An arc is given to connect two nodes if the two correspondences are consistent. Another major difference between IFG and LFF is in the task domain. The method by Bolles and Cain (1982) is for recognition of 2D objects (e.g., flat hinges) out of 2D images, while the IFG is for recognition of 3D objects from 2D images.

The index feature group is similar to Lowe's (1987) perceptual grouping, in which lines are grouped together by using the principles of proximity, parallelism, and collinearity. The grouped lines are used as indexing terms into models for reducing the matching effort. In his formulation, Lowe (1987) derived numerical measures of significance to account for the local density of features (or scale). It could be argued that IFG discussed here is a direct extension of Lowe's perceptual grouping of lines to that of conic sections without taking into account the local density of features.

2.3 Matching Image Primitives and Model Primitives

To recognize objects from an image is to establish consistent correspondences between image elements and model elements. To illustrate the computational complexity of the matching, consider the problem of establishing L consistent matches between I image elements and M model elements. Clearly, we have IxM possible pairs for the first correspondence, (IxM-1) possible pairs for the second correspondences, (IxM-2) for the third, and so forth. Thus, for this problem, the maximum number of $\begin{bmatrix} IxM \\ L \end{bmatrix}$ or roughly $(IxM)^L$ potential pairs are to be tried. This worst case is equivalent to searching every node in a L level tree, in which each level has IxM nodes.

In order to reduce the computational complexity, two simple approaches can be taken. One is to reduce the number of image elements or the number of model elements. This leads to the use of primitives (a cluster of elements). However, a primitive cannot be too large, or else it may not be reliable since it may include elements that do not belong to a detected primitive. Usually, the primitives are chosen to correspond to physical items that have identifiable physical properties; for example, corners, lines, and conics. The other way to reduce computational complexity is to use constraints or available knowledge to prune the fruitless branches in the search tree and order the matching sequence in promising branches. Three types of constraints, according to the taxonomy in Chapter 1, are be used. They are object-induced constraints, projection-induced constraints, and transformation-induced constraints.

The object-induced constraints are the properties and relations encoded in the object shape (note that surface reflectance, texture, and colour are not used here). The first step in applying object-induced constraints is to encode object primitives and their geometrical properties and relations in nodes and links of the stored models. Further, certain knowledge about an object is abstracted as rules and stored in the model for that object. The second step in using object-induced constraints is to retrieve the encoded knowledge to avoid unpromising paths and to order the matching sequence at different stages of matching.

The constraints from perspective projection are the inter-relations like the ones in Table 2.3. It is well known that, under perspective projection, regular conics are projected into regular conics, though not necessarily of the same type (Retorys, 1969). Table 2.3 is derived under the fact that the viewed conics are in front of the camera. A proof of Table 2.3 is given in Appendix A. The inter-relation between image primitives and model primitives can help initiate probable models to match a specific primitive in the image.

Another type of projection constraint is the index feature group discussed earlier. The underlying assumption for index feature groups is that, if certain primitives of the object appear in the image, then other neighboring primitives of the same object are likely to appear in the neighborhood of the identified image primitives. To illustrate the use of projection constraint in reducing matching effort, consider again the problem of matching I image primitives and M model primitives. Suppose a data-driven reasoning method is used; that is, image primitives are selected and then labelled as certain model primitives. If a most significant image primitive (an primitive most likely to have a counterpart in a certain object model) is always available at each level of the search tree, then the computational complexity of the problem can be reduced to $(M_{likely})^L$ instead of $(IxM)^L$, where M_{likely} is the number of likely model primitives to be paired with the selected significant image primitive. Here likely model primitives for a given image primitive are those model primitives satisfying some found constraints; e.g., if an image line is found to be parallel to another image line, the likely model primitives are model lines that are parallel to other model lines. The reason for the reduction in time is that the selection of significant image primitives avoids the need for trying image primitives that may have no counterparts in the model. In addition, a smaller number of model primitives are needed to be matched to the selected image primitive, as M_{likely} is generally much smaller than M.

Table 2.3 Projection of Conic Sections

Model primitives	Point	Line	Ellipse	Hyperbola	Parabola
Image primitives	Point	Line Point*	Ellipse Line*	Hyperbola line*	Ellipse Hyperbola Parabola Line*

Note.The object containing the conic sections is assumed to be in front of the camera. The image primitives marked with "*" are the degenerate cases which seldom occur.

The common fate principle of transformation constraints means that if a solution of object location (orientation and position) is found from the established correspondences, the solution should be valid for other correspondences that are consistent with the established ones. This principle is used with object-induced constraints to test if a set of correspondences is consistent, and to predict new correspondences. As indicated in Chapter 1, two ways can be taken to use the common fate principle. One is to propagate transformation constraints (formulated as inequalities) over orientation parameters and position parameters. The inequality, when orientation is parametrized by using quaternion (to be discussed in Chapter 3), is of quadratic form. Examples of these inequality constraints are given in Appendix B, and the closed-form algorithms for propagating quadratic constraints over its parameters are given in Appendix C. However, experiments with propagating constraints over intervals of q (the quaternion vector) and p (the position vector) indicated that a large number of correspondences have to be included to provide sufficient constraint to reduce the interval of q and p to a managable size. Also, a large number of correspondences have to be included to detect an inconsistency among them. This way of applying location information is found to be inefficient, at least when formulated as shown in Appendix B. The inefficiency increases with the number of unknown parameters. Thus, this application of the common fate principle was not used in implementing the proposed method.

Another way of using the common fate principle is to initially locate the object with a minimum number of correspondences (e.g., one ellipse, three lines, or three points), and then use the estimated object location to predict and establish new correspondences. This approach proves very efficient and was used in the implementation of the proposed method. To illustrate the efficiency of using the estimated location information, consider the matching problem for I image primitives and M model primitives again. Suppose the object location is available, and a model-driven reasoning is used; that is, model primitives are selected first and then labelled as certain image primitives. Since the object location is known, visible primitives are projected by using the ideal perspective model and the most significant model primitive (e.g., a model line having the longest length after projection) is selected and labelled as one of the image primitives in the neighborhood of the ideally projected primitive. As can be seen, the computational complexity can be reduced to $(I_{neighbor})^L$, where $I_{neighbor}$ is much smaller than I, because only image primitives near the predicted location in the image are tested, instead of primitives in the entire image.

Obviously, purely model-driven matching by using location information is not initially possible since the object's location is not known in advance. Therefore, at the initial stage, index feature groups are used to form hypotheses with a minimum number of correspondences. Thereafter, the location information from the minimum correspondences is used to predict and establish new correspondences. When additional correspondences are included, we update the estimation of object location. This, in turn, leads to more accurate prediction of other correspondences. Thus, less effort is needed to obtain correspondences at a deeper level of the search tree. In such a case, the complexity of matching I image primitives and M model primitives becomes $(M_{likely})^3(I_{neighbor})^{L-3}$, if the minimum number of correspondences is 3 (e.g., line or point correspondences). If one ellipse correspondence is found at the beginning, then the complexity becomes $(M_{likely})(I_{neighbor})^{L-1}$.

Having understood the saving in time of using significant features and location information, we now discuss details of the functions in the proposed matching method for object recognition.

Indexing Likely Models. Once significant feature groups are extracted as discussed in the previous section, they are represented in conjunctive normal form in predicate logic. For example, a group of two parallel lines is expressed as parallel(L1,L2) $\wedge$ line(L1) $\wedge$ line(L2), and a group of intersecting ellipse and line is represented as intersection(E1,L1) $\wedge$ line(E1) $\wedge$ line(L1). To retrieve likely models, the most significant feature group (one that can strongly indicate which object generates the features in the group and can provide strong location information of the object) is selected. The logic expression of the selected group is then used to match preconditions of the rules in the index table. Those rules with matched preconditions are triggered to index the likely models. Figure 2.1 in Section 2.1

illustrates this table-look-up operation. If no rules in the index table can be triggered, either a subset of the conjunctions of the selected feature group is used to have a relaxed preconditions for the table look-up or another feature group is selected.

Formulating Minimum Correspondence Hypothesis. As will be seen in Chapter 3, the minmum number of line or point correspondences needed to located an object in space is three, and that of elliptic or hyperbolic correpondences is one. Thus, a formulated hypothesis is required to have the minimum number of correspondences such that the hypothesis can be tested by the hypothesis-tester using location recovered from the obtained correspondences. When a likely model is initiated by the index table and its primitives are matched to the primitives in the index feature group, sufficient constraints may be available to locate the object in space. For example, when the feature group is of more than one ellipse, or three-line vertex, or four-line trapezoid. In such a case, correspondences are used directly to form a hypothesis and routed to the hypothesis-tester (to be discussed later in this section). However, in cases where correspondences are not sufficient to locate the object in space (e.g., a feature group of two near-parallel lines) an additional correspondence is needed to have the minimum number of correspondences. When this occurs, the symbolically expressed knowledge in the model is used to seek the most needed additional correspondence. For example, when two parallel line correspondences are obtained, the most needed line correspondence is the one whose model line is not parallel to the two matched model lines (since three parallel line correspondences are not sufficient in locating the object, as will be shown in Chapter 3). Hence, any line intersecting either of the matched lines in the model can be selected and matched to image lines which intersect the two near-parallel image lines. In this case, of course, several possible sets of minimum correspondences exist. These possible sets are treated as probable hypotheses to be tested by the hypothesis-tester. The operations required in formulating hypotheses are shown in Figure 2.4.

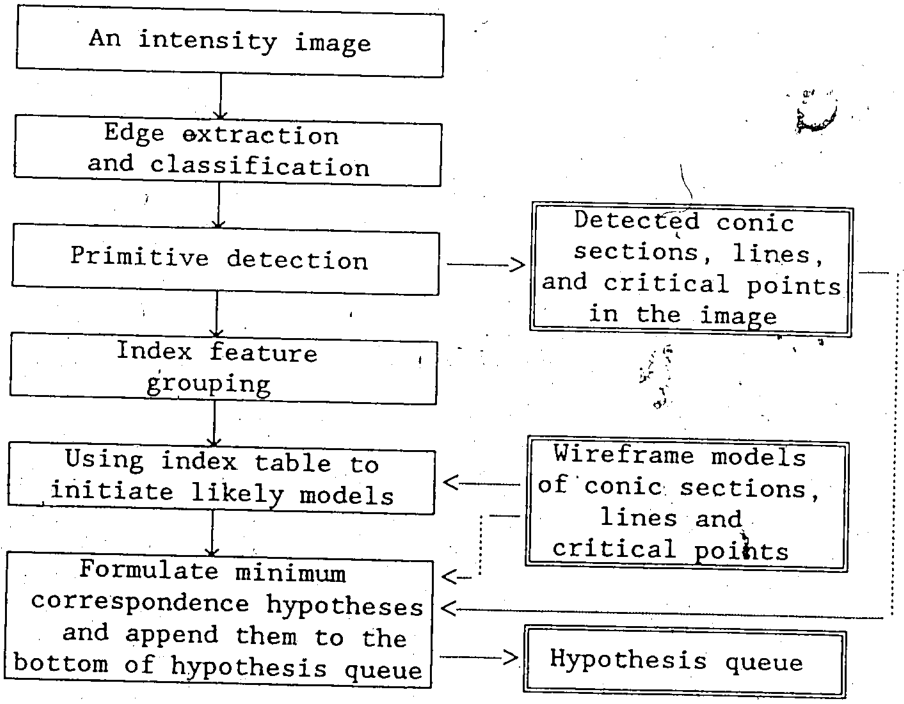

An intensity image
Edge extraction and classification
Primitive detection
Index feature grouping
Using index table to initiate likely models
Formulate minimum correspondence hypotheses and append them to the bottom of hypothesis queue
Detected conic sections, lines, and critical points in the image
Wireframe models of conic sections, lines and critical points
Hypothesis queue

Figure 2.4 Block diagram of the minimum-correspondence hypothesis-formulator. Single-line boxes are operations, and double-line boxes are datablocks or databases. The solid-line arrows are for control flow, while dashed-line arrows are for data flow.

<u>Testing Hypothesis</u>. When a hypothesis is routed to the hypothesis tester, the hypothesis contains sufficient correspondences to locate the object in space. Hence, the first step the hypothesis-tester takes is to call the locating modules (to be derived in Chapter 3) to

estimate the object pose from the obtained correspondences. Then it uses the recovered location to test if the obtained correspondences are mutually consistent (the consistency test is to be discussed later in this section). If inconsistency exists, the hypothesis-tester rejects the current hypothesis and takes on the next hypothesis in the job queue. If, however, the correspondences are found to be consistent, the hypothesis-tester calls the predicter to predict the visibility of other primitives of the initiated model by using the recovered object location. The predicter also generates predicted image attibutes of each visible primitive. For example, length and slope of predicted lines, and length and occupied area of predicted arc. These predicted attributes are used to evaluate the significance of predicted image primitives. For example, the longer an image line or arc is, the more significant it is. The model primitive with the most significant predicted image counterpart is selected to be matched to those detected image primitives whose measured attributes are close to the predicted attributes. For example, those image lines with slopes and occupied areas close to the predicted slope and occupied area are candidates to be paired to the selected model primitive. Working on these candidate image primitives with the selected model primitive, the hypothesis-tester incrementally augments the set of correspondences and then tests them by repeatedly calling the locating modules, consistent tester, and predictor. The cycle of localization, consistency testing, prediction, and augmentation continues until the set of consistent correspondences meets the criteria of confirmation in the initiated model or no possible consistent correspondences can be augmented. Figure 2.5 illustrates the operations involved in the hypothesis-testing cycle.

In testing consistency of correspondences, we define two predicates: attribute-fit() and area-fit(). The predicate, attribute-fit(d, $\hat{d}$, ϵ), is TRUE, if $\|d-\hat{d}\| \leq \epsilon$, and FALSE, otherwise. The argument, d, is a measured attribute of an image primitive, and $\hat{d}$ is the projected attribute of the corresponding model primitive using the recovered object location. The argument, ϵ, is the maximum tolerable error. The second predicate, area-fit($X_1, Y_1, X_2, Y_2, \hat{X}_1, \hat{Y}_1, \hat{X}_2, \hat{Y}_2, \varsigma, \xi$), is TRUE, if $([X_1-\varsigma, X_2+\varsigma] \cap [\hat{X}_1-\varsigma, \hat{X}_2+\varsigma]) \,/\, [X_1-\varsigma, X_2+\varsigma] \geq \xi$ and $([Y_1-\varsigma, Y_2+\varsigma] \cap [\hat{Y}_1-\varsigma, \hat{Y}_2+\varsigma]) \,/\, [Y_1-o, Y_2+o] \geq \xi$. Otherwise, it is FALSE. The arguments, X_1, Y_1, X_2, and Y_2 define the rectangle enclosing a detected image primitive, and $\hat{X}_1$, $\hat{Y}_1$, $\hat{X}_2$, and $\hat{Y}_2$ are their corresponding projected values using the recovered object location (see Figure 2.6). The argument, ς, is the offset for the defined rectangle, and ξ is the minimum required percentage of overlap between the measured rectangle and the projected rectangle. For convenience of discussion, the two predicates will be referred to with the name of the attribute or the two points defining the area instead of full arguments. For example, the fit of the slope of a line is expressed as at[illegible] of the line). Therefore, conditions for different kinds of primitives to be consistent with the estimated location are defined as follows:

An elliptic or hyperbolic correspondence is consistent with the recovered location if

attribute-fit(X of the center) $\wedge$ attribute-fit(Y of the center)

$\wedge$ attribute-fit(major semi-axis) $\wedge$ attribute-fit(minor semi-axis)

$\wedge$ area-fit(upper-left and lower-right corners of the enclosing window).

A line correspondence is consistent with the recovered object location if

attribute-fit(slope) $\wedge$ attribute-fit(Y-intercept)

$\wedge$ area-fit(two endpoints of the line).

A point correspondence is consistent with the recovered object location

attribute-fit(X of the point) $\wedge$ attribute-fit(Y of the point)

Chapter Summary. A new "locating-labelling paradigm" has been discussed in detail in this chapter. Three components in the proposed method are addressed. The first is the representation of object models. The second includes the techniques for detecting conic primitives in an intensity image and the representation of detected primitives. The third component contains the initial formulation of hypotheses and the testing of hypotheses. The hypothesis testing involves the cycling of localization from hypothesized matches, consistency testing of the matches with recovered location, the prediction of additional matches, and the augmentation of the set of hypothesized matches. The localization process from hypothesized matches is the most important part of the proposed "locating-labelling" method for object recognition. As such, all techniques addressed in this chapter are selected or devised in order to apply the locating algorithms in localization process and to exploit the information from this process. The algorithms for locating an object from correspondences of conic primitives will be discussed in the next chapter.

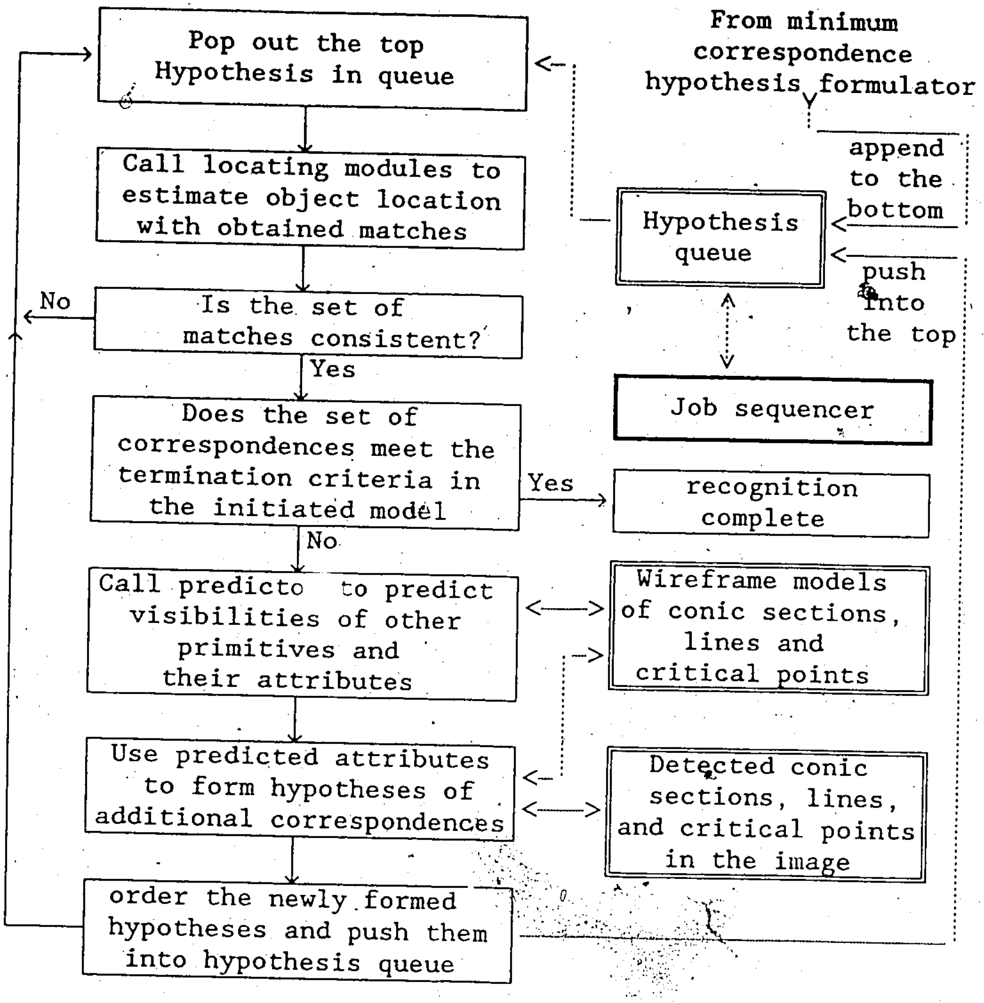

Pop out the top Hypothesis in queue
Call locating modules to estimate object location with obtained matches
No
Is the set of matches consistent?
Yes
Does the set of correspondences meet the termination criteria in the initiated model
Yes
No
Call predicto to predict visibilities of other primitives and their attributes
Use predicted attributes to form hypotheses of additional correspondences
order the newly formed hypotheses and push them into hypothesis queue
From minimum correspondence hypothesis formulator
append to the bottom
push into the top
Hypothesis queue
Job sequencer
recognition complete
Wireframe models of conic sections, lines and critical points
Detected conic sections, lines, and critical points in the image

Figure 2.5 Block diagram of the hypothesis-tester. Single-line boxes are operations, and double-line boxes are data blocks or database. The solid-line arrows are for control flow, while dash-line arrows are for data flow.

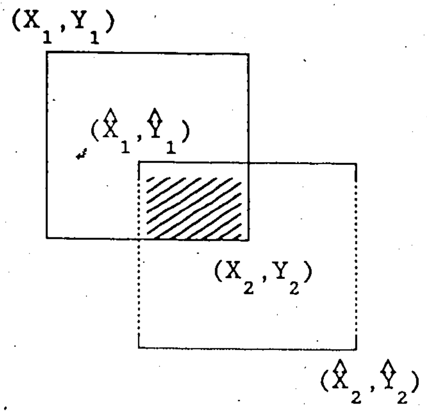


Figure 2.6 Illustration of area fit of a primitive. The solid line rectangle defined by (X_1, Y_1) and (X_2, Y_2), encloses the detected image primitive. The dashed line rectangle is the projected enclosing rectangle using the recovered location and the corresponding model primitive. The shaded area is the overlap of the detected and projected enclosing rectangles.

Chapter 3. Object localization

Recovery of object location is crucial in certain tasks like manipulation of viewed work pieces in manufacturing. For this purpose, it is often desirable to determine object location in 3D after the object has been recognized. In addition to its practical application, the ability to determine object location during the recognition process can greatly diminish the search effort required to match the given image and its originating object models, as shown in Chapter 2. Unfortunately, this latter use in image interpretation is often ignored, partially due to the difficulty and time complexity required to determine the 3D object location (e.g., a nonlinear Newton method is used by Lowe, 1987, to find the object location). To exploit localizability and fast localization methods, this chapter is devoted to providing various algorithms for determining object location using established correspondences. As will be shown, the minimum number of line or point correspondences required to determine object location is three, and that of elliptic or hyperbolic correspondences is one. Due to the fact that measurements on images are subject to noise, the accuracy of location estimation, in general, increases as the number of correspondences increases. Thus, the location estimation goes from coarse to fine as additional correspondences are assimilated.

In this chapter, the localization problem is categorized according to the number and type of correspondences and treated separately in each section.

Before we move on, it is necessary to review the methods for representing the orientation of the object relative to a specific coordinate basis. Although there are many ways to parameterize a rotation (Stuelpnagel, 1964), only four methods relevant to this thesis will be summarized. They are the nine-parameter expression, the roll-pitch-yaw expression, the quaternion expression, and the Rodrigues expression. The nine-parameter expression simply uses a 3x3 matrix to represent the rotation with the constraints that the rows and/or columns of the matrix are orthonormal. The roll-pitch-yaw expression uses three consecutive rotations around three principal axes to characterize a rotation:

$$O = RPY(\theta) \equiv Rot(z,\theta z)Rot(y,\theta y)Rot(x,\theta x)$$

$$= \begin{bmatrix} C\theta zC\theta y & C\theta zS\theta yS\theta x-S\theta zC\theta x & C\theta zS\theta yC\theta x+S\theta zS\theta x \\ S\theta zC\theta y & S\theta zS\theta yS\theta x+C\theta zC\theta x & S\theta zS\theta yC\theta x-C\theta zS\theta x \\ -S\theta y & C\theta yS\theta x & C\theta yC\theta x \end{bmatrix} \qquad (3.1)$$

where $\theta=[\theta x,\theta y,\theta z]^T$, $C\theta x=\cos(\theta x)$, $S\theta x=\sin(\theta x)$, $C\theta y=\cos(\theta y)$, $S\theta y=\sin(\theta y)$, $C\theta z=\cos(\theta z)$, and $S\theta z=\sin(\theta z)$. The operator Rot(axis,angle) is for the rotation with the specified angle around the specified axis.

The quaternion expression uses the rotation axis (around which the rotation takes place) and the rotation angle (the amount of rotation) to parameterize a rotation:

$$O = \begin{bmatrix} q_1^2-q_2^2-q_3^2+q_4^2 & 2(q_1q_2-q_3q_4) & 2(q_1q_3+q_2q_4) \\ 2(q_1q_2+q_3q_4) & -q_1^2+q_2^2-q_3^2+q_4^2 & 2(q_2q_3-q_1q_4) \\ 2(q_1q_3-q_2q_4) & 2(q_2q_3+q_1q_4) & -q_1^2-q_2^2+q_3^2+q_4^2 \end{bmatrix}$$

with the constraint that $q_1^2+q_2^2+q_3^2+q_4^2=1$.

In the quaternion expression, the vector $[q_1,q_2,q_3]^T$ defines the rotation axis and $2\sin^{-1}(q_4)$ indicates the rotation angle. The disadvantage of the quaternion expression is that it uses four parameters to represent a matrix with three degrees of freedom. Since a direction in 3D space has two degrees of freedom, one of the quaternion parameters can be eliminated. This results in the Rodrigues expression which uses three parameters for rotation axis and rotation angle to characterize a general rotation (Rosenberg, 1977). Let $s_1 = q_1/q_4$, $s_2 = q_2/q_4$, and $s_3 = q_3/q_4$, we have the Rodrigues expression for a rotation as follows:

$$O = (1+s_1^2+s_2^2+s_3^2)^{-1}\begin{bmatrix} 1+s_1^2-s_2^2-s_3^2 & 2(s_1s_2-s_3) & 2(s_1s_3+s_2) \\ 2(s_1s_2+s_3) & 1-s_1^2+s_2^2-s_3^2 & 2(s_2s_3-s_1) \\ 2(s_1s_3-s_2) & 2(s_2s_3+s_1) & 1-s_1^2-s_2^2+s_3^2 \end{bmatrix}$$

where the rotation axis is specified by $[s_1, s_2, s_3]^T$ and the rotation angle is given by $2\tan^{-1}(1/\sqrt{s_1^2+s_2^2+s_3^2})$.

Of these four types of expression, the nine-parameter expression and the roll-pitch-yaw expression are used in this chapter. The quaternion expression is used in Appendix B for constraint propagation. The Rodrigues expression is used in Section 3.3 to re-illustrate that the number of solutions for the localization problem with three line correspondences is four. When an Extended Kalman Filter (EKF) is used in Chapter 4, we use only the roll-pitch-yaw expression. The quaternion expression and the Rodrigues expression are avoided in that context, because the former requires more parameters than needed, and the latter has uneven distribution of value in parameters.

In the rest of the this chapter, the symbol O and p are for the orientation matrix and position vector, respectively. If necessary, the orientation matrix is parameterized with roll-pitch-yaw angles θ_z, θ_y, and θ_x. In all cases, the focal length of the camera is assumed to be known and its symbol is f. This assumption is practical because, even for an auto-focus camera, various focal lengths can be pre-calibrated and saved in a table for table-look-up retrieval. The camera is defined in a right-handed coordinate basis to have the X-axis lie on the horizontal axis and Y-axis on the vertical axis of the image. As a result, objects in front of the camera will have negative z-component in their relative position vector to the camera origin. Due to the difficulty of using unique symbols for different concepts, the symbols defined in each section are only valid for that section, unless specified otherwise.

3.1 One-Ellipse Localization (The E1 problem)

It is sufficient to determine the object location by using a correspondence between image and model ellipses. The proof and the procedure are given in Appendix D.1. As can be seen there, four possible locations exist for the E1 problem. If one face of the model ellipse is known to be visible while the other is not, only two of the four solutions are physically possible. The two solutions are shown in Figure 3.1 with the visible side facing the camera origin.

However, when the model ellipse is a circle, one correspondence is not sufficient to locate the object since any rotation around the axis passing through the center and perpendicular to the plane of the circle can not be recovered. In this case, we have 12 linear equations for 14 unknowns (see Appendix D). Thus, an additional line or point correspondence is needed to have 14 linear equations for the 14 unknowns. The two equations from the additional line or point will be derived later, when we discuss line and point correspondences in Section 3.3 and Section 3.5. Alternatively, see equations (D.2.1), (D.2.2), (D.3.1) and (D.3.2) in Appendix D.

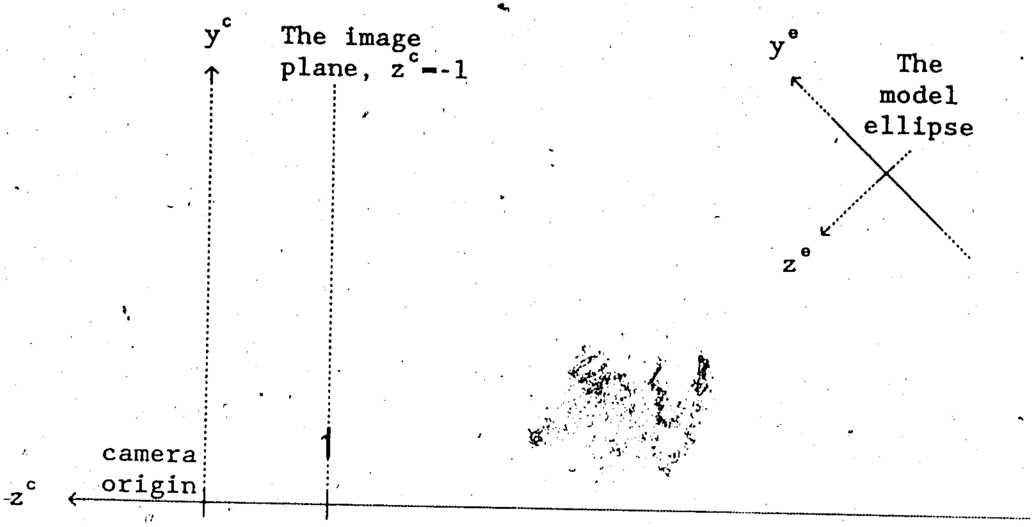


Figure 3.1 The perspective projection of an ellipse defined on the plane $z^e = 0$. The dashed lines are for the axes of coordinate bases. The solid segment is for the ellipse sliced by the plane $x^e = 0$. Let x^e be aligned with x^c, the image of the ellipse is the bold segment in the image plane. On the other hand, when the image ellipse (i.e., the bold segment) is given, there are two possible orientations for the model ellipse to produce the given image ellipse if the model ellipse can only be seen from the side of positive z^e. One orientation solution is the transformation of the solid segment to the camera origin, and the other is the 180 degree rotation of the first solution around the z^e axis.

3.2 n-Ellipse Localization (The En problem, where $n > 1$)

When more than one ellipse correspondence is available, one has a set of 12 equations, from Appendix D.1, for each correspondence:

$$O\,Q = G \tag{3.1}$$

$$O\,s + p = t \tag{3.2}$$

where Q and s specify the transformation between the conic primitive and the model coordinate basis; they are known and stored in the model. G and t specify the transformation between the conic primitive and the camera; they are obtained by using the *ElLoc* algorithm in Appendix D.1.

O and p are the transformation from model coordinate basis to the camera coordinate basis; the localization problem involves the determination of O and p.

The problem of determining O and p is now over-constrained, since we have 12xn equations for 9 parameters in O and 3 parameters in p (an over-constrained solution is usually desirable for smoothing out the noise effect). Because the equations are linear in the unknown parameters, an unique solution for the unknowns can be determined by using a pseudo-inverse method, in which the vector $(A^TA)^{-1}(A^Td)$ is the solution for the over-constrained linear equation (Ballard and Brown, 1982)

$$A\,b = d \qquad (3.3)$$

where b is the unknown vector, and A and d are known constant matrix and vector.

3.3 Three-Line Localization (The L3 problem)

The algorithm for locating the object by using three line correspondences is derived in Appendix D.2. The condition for the three line locator, *L3Loc*, to work is that the three lines are not all parallel and any two of the three lines are not collinear. The maximum number of possible solutions from *L3Loc* for O and p are four. Since the algorithm is to solve O and p such that the projected model lines have the same slopes and Y-intercept as their image conterparts, regardless of their occupied regions, those of the four solutions for O and p that produce unmatched regions between the projected model lines and image lines can be eliminated.

It is worth mentioning tHat the slopes of image lines are assumed to be finite in the derivation in Appendix D.2. In the case of a vertical image line, the inverse of line slope and X-intercept, instead of slope and Y-intercept, are used to described the image line. However, the derivation in Appendix D.2 is still valid for the mixture of these two kinds of image line descriptions.

It is not surprising to see that there are other ways, than the one given in Appendix D.2, to solve the L3 problem. One other way is to parameterize the O matrix with Rodrigues parameters, s_1, s_2, and s_3. From (D.2.1) in Appendix D, we have, for three line correspondences, three quadratic equations:

$$\mathbf{b}_1^T \, O \, \mathbf{a}_1 = [s_1,s_2,s_3,1] \; A_1 \; [s_1,s_2,s_3,1]^T = 0$$
$$\mathbf{b}_2^T \, O \, \mathbf{a}_2 = [s_1,s_2,s_3,1] \; A_2 \; [s_1,s_2,s_3,1]^T = 0$$
$$\mathbf{b}_3^T \, O \, \mathbf{a}_3 = [s_1,s_2,s_3,1] \; A_3 \; [s_1,s_2,s_3,1]^T = 0$$

where $\mathbf{b}_i^T = [-m_i f, \; f, \; T_i]/\|[-m_i f, \; f, \; T_i]\|$, and m_i and T_i are the slope and Y-intercept for image line i. The unit vector $\mathbf{a}_i$ specifies the direction of the model line of the i-th correspondence in the model coordinate basis. A_1, A_2, and A_3 are 4x4 symmetric matrices.

It can be seen that there are, at most, eight solutions for the intersections of the three quadrics. One way to find intersections of quadrics can be found in the works by Levin (1976 and 1979) and Sarraga (1983). Once s_1, s_2, and s_3 are found, we can then solve the position vector as:

$$\mathbf{p} = \begin{bmatrix} \mathbf{b}_1^T \\ \mathbf{b}_2^T \\ \mathbf{b}_3^T \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{b}_1^T \, O \, \mathbf{r}_1 \\ \mathbf{b}_2^T \, O \, \mathbf{r}_2 \\ \mathbf{b}_3^T \, O \, \mathbf{r}_3 \end{bmatrix}$$

where $\mathbf{r}_i$ is any point on the model line, i, in the model coordinate basis.

Of the eight possible solutions, only four of them have a negative z-component in the recovered position vector, **p**, to have the object in front of the viewing camera.

3.4 n-Line Localization (The Ln problem, where $n > 3$)

If $n \geq 6$, we have at least 12 equations like (D.2.1) or (D.2.2) in Appendix D.2. This is an over-constrained problem, and a unique solution for O and p can be found by using the Pseudo-inverse method used in Section 3.2.2.

If $n = 4$ or $n = 5$, multiple solutions for O and p may exist. An approach to tackle this problem can be the following two-step algorithm. First, three of the correspondences are used by the *L3Loc* algorithm to obtain a maximum number of four solutions. Second, each solution is then used as an initial guess in a two-parameter *IEKF* algorithm (to be derived later in Appendix E) to find the solution that minimizes the weighted mean-square-error of the 2xn equations given by the n correspondences.

The three correspondences selected in the first step are those correspondences whose image lines have more reliable measurements (i.e., slope and Y-intercept) than others. For example, longer image lines provide more reliable slope measurement that shorter lines.

In the second step, there are $2 + 2 \times (n - 3)$ equations for the *IEKF* algorithm. The first two equation are (D.2.9) and (D.2.10) in Appendix D, which are duplicated as follows:

$$\mathbf{b}_2^T G\ Q\ F\ \mathbf{a}_2 = 0 \tag{3.4}$$

$$\mathbf{b}_3^T G\ Q\ F\ \mathbf{a}_3 = 0 \tag{3.5}$$

where $\mathbf{b}_i^T = [-m_i f,\ f,\ T_i]/\|[-m_i f,\ f,\ T_i]\|$, and m_i and T_i are the slope and Y-intercept for image line i. The unit vector $\mathbf{a}_i$ specifies the direction of the model line of the i-th correspondence in the model coordinate basis. The matrices F and G are chosen such that they satisfy (D.2.4)-(D.2.7). Finally, the matrix Q is defined as $G^T O F^T$.

Each additional correspondence, j (=4 or 5), gives two additional equations:

$$b_j^T G\ Q\ F\ a_j = 0 \tag{3.6}$$

$$b_j^T G\ Q\ F\ r_j - b_j^T \begin{bmatrix} b_1^T \\ b_2^T \\ b_3^T \end{bmatrix}^{-1} \begin{bmatrix} b_1^T\ G\ Q\ F\ r_1 \\ b_2^T\ G\ Q\ F\ r_2 \\ b_3^T\ G\ Q\ F\ r_3 \end{bmatrix} = 0 \tag{3.7}$$

where a_i and b_i are defined in (3.5), and r_i is any point on the model line, i, in the model coordinate basis.

Since the pitch angle, φ_y, in the Q matrix is zero from (D.2.8), there are only two unknown parameters, φ_x and φ_z, in the Q matrix to be estimated. In using the *IEKF* algorithm or any other optimization algorithm, equation (3.6) and (3.7) should be weighted differently since a_j is an unit vector while r_j is in real distance (e.g., in centimeters). As in any gradient method, derivatives of (3.4)-(3.7) relative to φ_x and φ_z are required in the *IEKF* algorithm. Generic forms of derivatives of measurement equations relative to roll-pitch-yaw angles for line or point correspondence are derived in Section 4.3. Once φ_x and φ_z are obtained, we can calculate Q and O (=GQF). Subsequently, we recover the position vector as:

$$p = \begin{bmatrix} b_1^T \\ b_2^T \\ b_3^T \end{bmatrix}^{-1} \begin{bmatrix} b_1^T\ O\ r_1 \\ b_2^T\ O\ r_2 \\ b_3^T\ O\ r_3 \end{bmatrix}$$

3.5 Three-Point Localization (The P3 problem)

Although the P3 problem can be solved by converting it into a L3 problem, closed form solutions via finding roots of a fourth-order algebraic equation are derived in Appendix D.3. Each real root of the fourth-order equation corresponds to a set of solution for O and p; thus, the maximum number of solutions for P3 problem is four. The independent result for the P3 problem mentioned above is the same as the one derived by Bolles and Fishchler (1981) to locate objects using only point correspondences.

3.6 n-Point Localization (The Pn problem, where n > 3)

It is clear that, when $n \geq 3$, the Pn problem can be converted into a Ln problem by using the n independent lines (or edges) which connect the given points. However, in some cases (e.g., $n \geq 6$ and $n = 3$), it may be more convenient to solve the Pn problem directly than to solve its corresponding Ln formulation. For $n \geq 6$, more than 12 equations like (D.3.1) or (D.3.2) are available; consequently, the pseudo-inverse method can be used to obtain a unique solution for O and p. In the case where $n = 4$ or $n = 5$, the Pn problem is transformed into a Ln problem and solved accordingly.

3.7 Two-Line-One-Point Localization (The L2P1 problem)

If the given point is not at the intersection of the two given lines, the L2P1 problem can be solved in a way similar to the L3 problem. Suppose the two model lines are described by $(\mathbf{a}_i, \mathbf{r}_i)$ and the two image lines by $(\mathbf{m}_i, T_i)$, where $i = 1,2$. The symbols $\mathbf{a}_i$, $\mathbf{r}_i$, $\mathbf{m}_i$ and T_i are defined in (3.5) and (3.7). Additionally, suppose the model point is given by $\mathbf{r}_3$ and the image point by (X_3, Y_3). Following Appendix D.2, one has

$$p = \begin{bmatrix} \mathbf{b}_1^T \\ \mathbf{b}_2^T \\ \mathbf{u}_3^T \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{b}_1^T O\, \mathbf{r}_1 \\ \mathbf{b}_2^T O\, \mathbf{r}_2 \\ \mathbf{u}_3^T O\, \mathbf{r}_3 \end{bmatrix} \qquad (3.8)$$

$$\mathbf{b}_1^T O\, \mathbf{a}_1 = 0 \qquad (3.9)$$

$$\mathbf{b}_2^T O\, \mathbf{a}_2 = 0 \qquad (3.10)$$

$$\mathbf{v}_3^T O\, \mathbf{r}_3 - \mathbf{v}_3^T \begin{bmatrix} \mathbf{b}_1^T \\ \mathbf{b}_2^T \\ \mathbf{u}_3^T \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{b}_1^T O\, \mathbf{r}_1 \\ \mathbf{b}_2^T O\, \mathbf{r}_2 \\ \mathbf{u}_3^T O\, \mathbf{r}_3 \end{bmatrix} = 0 \qquad (3.11)$$

where $\mathbf{b}_i^T = [-m_i f,\ f,\ T_i] / \|[-m_i f,\ f,\ T_i]\|$, $\mathbf{u}_3^T = [f,\ 0,\ X_3]/\|[f,0,X_3]\|$, and $\mathbf{v}_3^T = [0,\ f,\ Y_3] / \|[0,\ f,\ Y_3]\|$. Note that the first matrix in (3.8) is invertible if the given image point does not lie on the intersection of the two image lines

Using the similar technique as in Appendix D.2, the O matrix can be obtained from (3.9)-(3.11). Subsequently, p can be calculated by using (3.8).

3.8 One-Line-Two-Point Localization (The L1P2 problem)

If one of two given points is not on the given line, the L1P2 problem can be converted into the L2P1 problem to have two lines (the given line and the line passing the two given points) and a point (whichever of the two points is not on the given line).

3.9 k-Line-n-Point Localization (The LkPn problem, where k+n > 3)

If k+n ≥6, the LkPn problem is over-constrained and can be solved by the pseudo-inverse method. If k+n < 6 and n ≥ 3, the LkPn problem can be converted to the L(k+n) problem and solved accordingly. If k+n < 6 and n < 3, the LkPn problem can be converted to the L(k+n-1)P1 problem which can be solved by the *L2P1Loc* algorithm and a two-parameter EKF algorithm similar to the one in Section 3.3.4.

3.10 j-ellipse-k-Line-n-Point Localization (The EjLkPn problem)

If $j \geq 1$, or if $k+n \geq 6$, the problem is over-constrained and can be solved by using the pseudo-inverse method. The remaining cases of EjLkPn problem are covered in previous sections.

Chapter Summary. In this Chapter, we have discussed extensively the problem of object localization using correspondences of conic primitives. It has been shown that three line/point or one elliptic/hyperbolic correspondence is sufficient to locate an object in 3D with the maximum of four possible solutions. Additional correspondences to the minimum number of matches required can provide a unique solution and improve the accuracy of the estimated location. The knowledge about the minimum number of (point, line, or ellipse) correspondences required for locating an object in 3D is used in Chapter 2 to formulate a minimum correspondence hypothesis. The ability to determine the 3D object location early (i.e., using minimum number of correspondences) in the object recognition process is shown in Chapter 2 to be advantageous. When additional correspondences are established, the estimation of the object location is refined by using the algorithms outlined in this chapter. The algorithms presented here cover all cases of the localization problem using conic correspondences. In addition to their use in object localization, the correspondences of conic primitives can also be used to recover the 3D object motion, which is to be addressed in the next chapter.

Chapter 4. Estimation of Motion Dynamics Using an Extended Kalman Filter

As mentioned earlier, the location and motion information is desired in planning trajectory and grasp formation for manipulators. In addition, the information can be used to verify the recognition of object identities. The main purpose of this chapter is to address this problem by modeling the motion dynamics and developing a method to estimate its parameters. The dynamic model used here assumes constant velocity motion, though higher order dynamics can be appended to it if so desired. The parameters in the dynamics are six unknowns for location (orientation and position at a specific time) and six unknowns for motion (rotation and translation between two sampling instants).

The problem of recovering these unknowns is formulated in the Extended Kalman Filter (EKF) format, in which the measured primitive attributes in the image along with models of motion dynamics and projection are used to estimate the unknowns in order to minimize an averaged mean-square-error at each frame. This chapter is organized as follows. Section 4.1 introduces the concept of EKF. The dynamic and measurement models are derived in Section 4.2 and Section 4.3, respectively. Since each object has its own motion parameters, each object can be treated independently. Thus, the discussion below assumes a single object.

4.1 The Extended Kalman Filter (EKF)

The EKF formulation has three essential concepts: the aspect of measured and predicted attributes, the aspect of dynamic and measurement models to approximate the nature of motion and camera projection, and the aspect of minimum-risk estimation. These concepts are illustrated in Figure 4.1 and are to be discussed below.

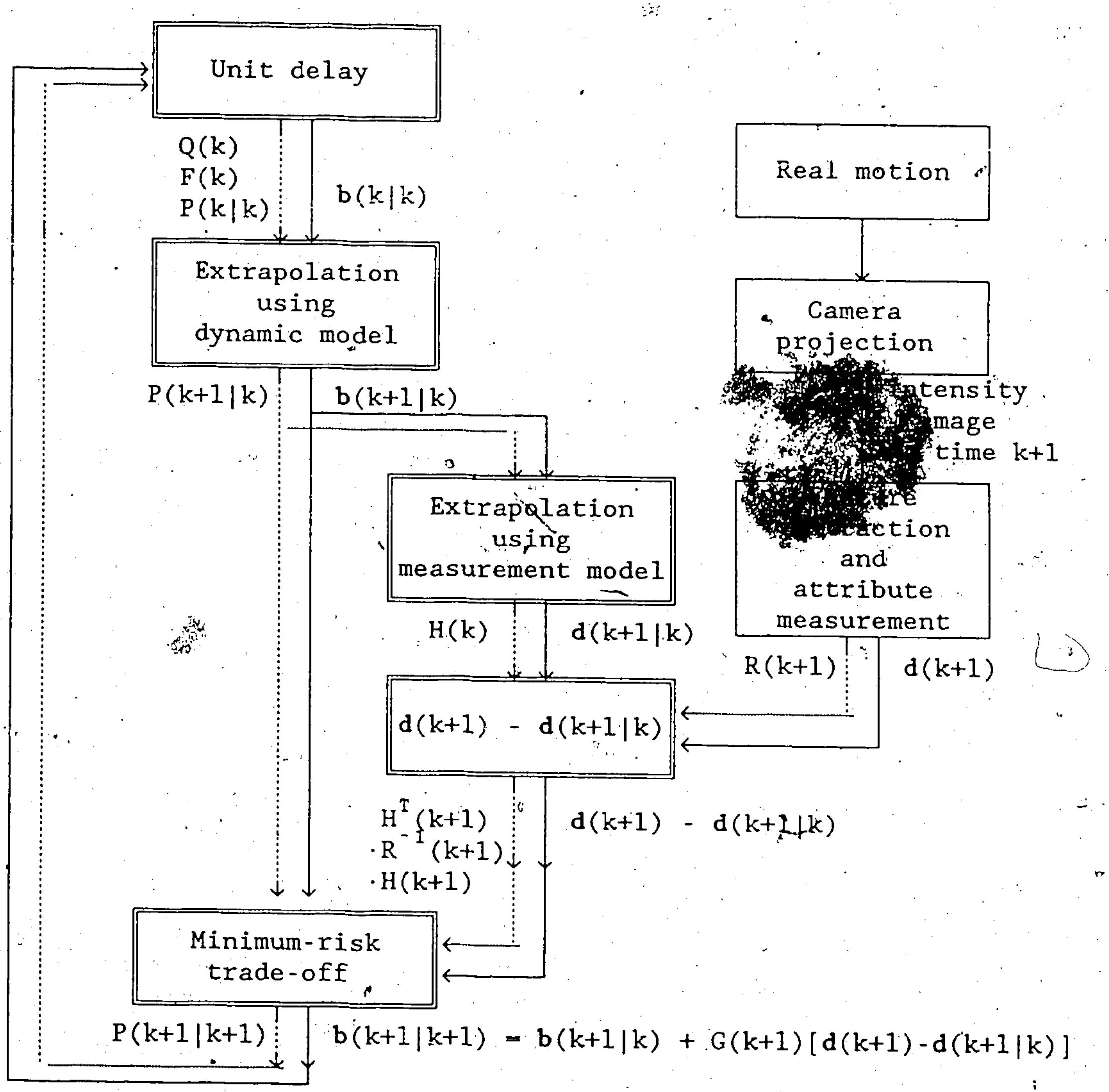


Figure 4.1 The block diagram of motion recovery using extended Kalman Filter (EKF) approach. The double-line boxes are for internal processes in the EKF, while single line boxes are for external processes. The solid-line arrow is for passing attributes and parameters, and the dashed-line arrow is for passing covariance matrices. The symbols are defined in Table 4.1.

Table 4.1. The symbols used in the EKF formulation.

$d(k)$	This measurement obtained at the time instant k.
$b(k)$	This vector contains 6 location parameters and 6 motion parameters at time instant k.
$n(k)$	This random vector accounts for the unmodelled dynamics at the time instant k.
$v(k)$	This random vector accounts for the measurement noise at the time instant k.
$b(k \mid k)$	The estimate of $b(k)$, conditioned on the measurement up to time instant k.
$b(k+1 \mid k)$	The estimate of $b(k+1)$, conditioned upon the measurement up to time instant k. It is obtained by extrapolating $b(k \mid k)$ with the known dynamic model.
$d(k+1 \mid k)$	The extrapolated measurement for time instant k+1, conditioned upon the measurement up to time instant k. It is obtained by extrapolating $b(k+1 \mid k)$ with the known measurement model.
$Q(k)$	Covariance matrix of $n(k)$.
$R(k)$	Covariance matrix of $v(k)$.
$P(k \mid k)$	Covariance matrix of $b(k \mid k)$.
$P(k+1 \mid k)$	Covariance matrix of $b(k+1 \mid k)$.
$F(k)$	The derivative of the vector $b(k+1 \mid k)$ with respect to the vector $b(k \mid k)$.
$H(k+1)$	The derivative of the vector $d(k+1 \mid k)$ with respect to the vector $b(k+1 \mid k)$.
$G(k+1)$	A matrix with relates the amount of update in the estimate $b(k+1 \mid k)$ to the difference between the measurement $d(k+1)$ and the extrapolated measurement $d(k+1 \mid k)$. In this formulation, $G(k+1) = P(k+1 \mid k+1)\, H^T(k+1)\, R^{-1}(k+1)$.

Measured Primitive Attributes. Given a camera image of viewed objects, primitives such as maximum curvature point, lines, conics, blobs, surfaces, and texture patterns can be identified on the image. Consequently, the attributes of these image primitives can be measured. The attributes can be the X-Y coordinates of a point, slope and Y-intercept of a line, or other properties for a shape or texture. Clearly, the attributes of these primitives can be measured by working on the given image alone. Generally speaking, these measured attributes can be obtained without knowing the nature of motion and projection and the shapes or other properties of the viewed objects. For convenience, the measured attributes at frame k are captured in the vector d(k) in the rest of this chapter. In EKF terminology, d(k) is called the measurement at time k.

Predicted Primitive Attributes. In contrast to measured attributes, if knowledge (whether it is preprogrammed, acquired, or hypothesized) about the viewed object shapes, the motion dynamics, and the projection is available, primitive attributes on an image at time k can be predicted before d(k) is measured and processed. For convenience, $d(k_2|k_1)$ is used to represent the predicted primitive attributes at time k_2, conditioned upon the entire history of measurements up to time k_1. In EKF terminology, $d(k_2|k_1)$ is called the extrapolated measurement for time k_2, conditioned upon measurements up to k_1. It should be noted that in predicting the attributes for the current frame before the they are actually measured and processed, the identity of the object is assumed to have been recognized in the previous frame. To predict the attributes, the 3D location of the object at the current frame time is predicted and then the object is perspectively projected using the predicted location. To predict the object location, we need to have the model of motion dynamics which relates location and motion parameters at one instant of time to those parameters at other instants of time. To project the object, we need to know the measurement model which relates the attributes of the image primitives to the location and motion parameters of the object. The dynamic model and the measurement model, in general, are:

$$b(k+1) = f(b(k)) + n(k) \tag{4.1}$$

$$d(k+1) = h(b(k+1)) + v(k+1) \tag{4.2}$$

In (4.1) and (4.2), the vector, b(k), contains 6 location parameters and 6 motion parameters. The vector function, f(*), relates the dynamically changing parameters in b(*) at different times. The vector, d(k+1), contains the measured attributes at time k+1. The vector function, h(*), relates the measured attributes in d(*) to the unknown parameters in b(*) at a particular time. n(k) in (4.1) is a random vector which accounts for the unmodelled dynamics and is assumed to be uncorrelated and with zero-mean Gaussian distribution. Similarily, v(k) in (4.2) is a random vector which accounts for noise both from the measurement of attributes and from the approximation of camera projection by perspective projection. It is also assumed to be uncorrelated and with zero-mean Gaussian distribution.

Having modelled the motion dynamics and the projection, we can then use the estimate of b(k), conditioned upon the history of measurement up to time k, to calculate the extrapolated parameters in b(k+1|k) and extrapolated attributes in d(k+1|k) by using the next two equations derived from (4.1) and (4.2):

$$b(k+1|k) = f(b(k|k)) \tag{4.3}$$

$$d(k+1|k) = h(b(k+1|k)) \tag{4.4}$$

<u>Minimum</u> <u>Risk</u> <u>Estimation</u>. After obtaining the extrapolated parameters in b(k+1|k), the extrapolated attributes in d(k+1|k), and the measured attributes in d(k+1), we can proceed to estimate the unknown parameters at time k+1, conditioned upon the entire history of measurements up to time k+1. This estimate is denoted by b(k+1|k+1). To obtain this estimate, we have to decide how much we trust (d(k+1)-d(k+1|k)) and how confident we are in the accuracy of the extrapolated vector b(k+1|k). The confidence measure of each vector is

expressed in terms of the covariance matrix of that vector. Conceptually, the one with larger covariance will receive harsher penalty. By propagating the the mean values and covariance matricies of $(d(k+1)-d(k+1|k))$ and $b(k+1|k)$, the optimal estimate (i.e., the one with minimum uncertainty measure) of the unknown vector $b(k+1)$ turns out to be one with the minimum covariance of $b(k+1|k+1)$.

A derivation of generic EKF is given in Appendix E. The algorithm *DynamicEKF* is for nonlinear dynamic systems such as the problem at hand. The algorithm *StaticEKF* is for nonlinear static systems such as the problem of recovering static object location by using images from a moving camera with known movement. The algorithm *IEKF* is used in the inner loop of *StaticEKF and DynamicEKF*. The algorithm *IEKF* can also be used in an unconstrained optimization problem such as the one required in the L4 and L5 problems in the previous chapter. It should be noted that the dynamic and measurement models, i.e., the vector functions $f(*)$ and $h(*)$ in (4.1) and (4.2), will vary from one problem to another, and have to be derived separately for each individual problem. The next two sections discuss the details of dynamic and measurement models for the problem of recovering motion parameters.

4.2 The Dynamic Model

In this section, we focus on the details of the vector function $f(*)$ in (4.1), that is, we obtain the dynamic model for the problem of motion recovery. Assuming that the rigid motion is of constant velocity, and that the object coordinate basis is at the center of mass of the object, we have the dynamic equations:

$$O(k+1) = O(k)\Delta O(k) \tag{4.5.a}$$

$$p(k+1) = p(k) + \Delta p(k) \tag{4.5.b}$$

$$\Delta O(k+1) = \Delta O(k) \tag{4.5.c}$$

$$\Delta p(k+1) = \Delta p(k) \tag{4.5.d}$$

where O(k) and p(k) are the orientation matrix and position vector of the object at time k. ΔO(k) and Δp(k) are the rotation matrix and translation vector for the motion between time k and k+1.

Although O(k) is a matrix with nine elements, it is governed by only the three parameters θ_x, θ_y, and θ_z in roll-pitch-yaw expression. Likewise, ΔO(k) is governed by $\Delta\theta_x$, $\Delta\theta_y$, and $\Delta\theta_z$. Let $\theta(k) = [\theta_x(k), \theta_y(k), \theta_z(k)]^T$ and $\Delta\theta(k) = [\Delta\theta_x(k), \Delta\theta_y(k), \Delta\theta_z(k)]^T$. We have, for the problem of motion recovery, b(k) in (4.1) as a composite vector of $\theta(k)$, p(k), $\Delta\theta(k)$, and Δp(k). Since $\theta(k+1)$ is independent of p(k) and Δp(k), in order to find the vector function, f(*), for (4.5), we need to express $\theta(k+1)$ only in terms of $\theta(k)$ and $\Delta\theta(k)$.

Recall the roll-pitch-yaw expression for O(k+1):

$$O(k+1) = RPY(\theta(k+1)) \equiv Rot(z, \theta_z(k+1))Rot(y, \theta_y(k+1))Rot(x, \theta_x(k+1))$$

$$= \begin{bmatrix} C\theta_z C\theta_y & C\theta_z S\theta_y S\theta_x - S\theta_z C\theta_x & C\theta_z S\theta_y C\theta_x + S\theta_z S\theta_x \\ S\theta_z C\theta_y & S\theta_z S\theta_y S\theta_x + C\theta_z C\theta_x & S\theta_z S\theta_y C\theta_x - C\theta_z S\theta_x \\ -S\theta_y & C\theta_y S\theta_x & C\theta_y C\theta_x \end{bmatrix} \tag{4.6}$$

where $C\theta_x = \cos(\theta_x(k+1))$, $S\theta_x = \sin(\theta_x(k+1))$, $C\theta_y = \cos(\theta_y(k+1))$, $S\theta_y = \sin(\theta_y(k+1))$, $C\theta_z = \cos(\theta_z(k+1))$, and $S\theta_z = \sin(\theta_z(k+1))$.

Combining (4.5.a) and (4.6) yields:

$$RPY(\theta(k+1)) = RPY(\theta(k))\ RPY(\Delta\theta(k)) \tag{4.7}$$

Let RPY^{-1} be the inverse operator of RPY. That is, RPY^{-1} takes an orientation matrix as the operand and returns its roll-pitch-yaw angles, whereas RPY takes roll-pitch-yaw angles as operands and returns the corresponding orientation matrix. Applying RPY^{-1} to both sides of

(4.7) yields:

$$\theta(k+1) = RPY^{-1}(RPY(\theta(k))\ RPY(\Delta\theta(k))) \qquad (4.8)$$

With (4.8) for O(k+1) and $\Delta\theta(k+1)$ for $\Delta O(k)$, (4.5) can be rewritten, to include unmodelled dynamics, as follows:

$$\theta(k+1) = RPY^{-1}(RPY(\theta(k))\ RPY(\Delta\theta(k))) + n_1(k) \qquad (4.9.a)$$

$$p(k+1) = p(k) + \Delta p(k) + n_2(k) \qquad (4.9.b)$$

$$\Delta\theta(k+1) = \Delta\theta(k) + n_3(k) \qquad (4.9.c)$$

$$\Delta p(k+1) = \Delta p(k) \text{ [illegible]} \qquad (4.9.d)$$

where $n_1(k)$, $n_2(k)$, n[illegible] and $n_4(k)$ account for the unmodelled dynamics.

As can be seen, (4.9) is the explicit expression for (4[illegible] with $b(k)=[\theta^T(k),p^T(k),\Delta\theta^T(k),\Delta p^T(k)]^T$ and $n(k)=[n_1^T(k),n_2^T(k),n_3^T(k)$ [illegible]$]^T$. The only thing lacking now is the explicit expression of RPY^{-1}, which is considered next.

<u>Details</u> <u>of</u> <u>inverse</u> <u>RPY</u>. Note that O(k+1) is an explicit function of O(k) and $\Delta O(k)$ which in turn are explicit functions of $\theta(k)$ and $\Delta\theta(k)$. In order to make explicit (4.9.a) in terms of $\theta(k)$ and $\Delta\theta(k)$, we express $\theta(k+1)$ in terms of elements of O(k+1). By doing this, we explicate RPY^{-1}. Let $o_{ij}(k+1)$ be the the (i,j) element in the matrix O(k+1). We solve for $\theta_z(k+1)$, $\theta_y(k+1)$, and $\theta_x(k+1)$ in terms of elements of O(k+1). Since the pitch angle, $\theta_y(k+1)$, lies between $-\pi/2$ and $\pi/2$, $o_{11}(k+1)$ has the same sign as $\cos(\theta_z(k+1))$; similarly, $o_{21}(k+1)$ has the same sign as $\sin(\theta_z(k+1))$. Thus, from the elements (1,1) and (2,1) on both sides of (4.6), we have:

$$\tan(\theta_z(k+1)) = o_{21}(k+1)/o_{11}(k+1)$$

which then yields:

$$\theta z(k+1) = \mathrm{ATAN2}(o_{21}(k+1), o_{11}(k+1)) \qquad (4.10)$$

where ATAN2(y,x) is a function which calculates arc tangent of y/x. The range of the function is from $-\pi$ to π. This arc-tangent function can be found in programming languages such as C and FORTRAN.

The exception of (4.10) is the degenerate case where $o_{11}(k+1) = o_{21}(k+1) = 0$. In this particular case, the angle $\theta z(k+1)$ is set to zero.

Next, to obtain $\theta y(k+1)$, note that $\cos(\theta y(k+1))$ is non-negative in the roll-pitch-yaw expression. Thus,

$$\cos(\theta y(k+1)) = (o_{21}^2(k+1) + o_{11}^2(k+1))^{1/2}$$

which is combined with the (3,1) elements in the matrices on both sides of (4.6) to produce:

$$\theta y(k+1) = \mathrm{ATAN}(-o_{31}(k+1), (o_{21}^2(k+1) + o_{11}^2(k+1))^{1/2}) \qquad (4.11)$$

where ATAN(x) is another arc tangent function whose range is from $-\pi/2$ to $\pi/2$. This function can also be found in C and FORTRAN.

Again, for the degenerate case where $o_{11}(k+1) = o_{21}(k+1) = 0$, $\theta y(k+1)$ is equal to zero.

Finally, for $\theta x(k+1)$, we use the elements (3,2) and (3,3) in the matrices on both sides of (4.6) and the fact that $\cos(\theta y(k+1))$ is nonnegative to obtain:

$$\tan(\theta x(k+1)) = o_{32}(k+1)/o_{33}(k+1)$$

which in turn yields:

$$\theta x(k+1) = \mathrm{ATAN2}(o_{32}(k+1), o_{33}(k+1)) \qquad (4.12)$$

where ATAN2(*) is defined in (4.10).

Again, for the degenerate case where $o_{32}(k+1) = o_{33}(k+1) = 0$, $\theta x(k+1)$ is set to zero.

<u>Calculating the Derivative of f(b(k|k)) with respect to b(k|k).</u> When using EKF, the derivative of f(b(k|k)) with respect to b(k|k) is required. The derivative is in fact a matrix, which is denoted by F(k) in Appendix E. Since the derivatives of (4.9.b)-(4.9.d) with respect to b(k) are trival, here we will only concern ourselves with $\partial RPY^{-1}(RPY(\theta(k|k))\ RPY(\Delta\theta(k|k)))/\partial b^T(k|k)$. To calculate this, the derivatives of RPY and RPY^{-1} have to be computed first. From (4.6), we have

$$\partial RPY(\theta)/\partial\theta_x = \begin{bmatrix} 0. & C\theta_zS\theta_yC\theta_x+S\theta_zS\theta_x & C\theta_zS\theta_yS\theta_x+S\theta_zC\theta_x \\ 0. & S\theta_zS\theta_yC\theta_x-C\theta_zS\theta_x & S\theta_zS\theta_yS\theta_x-C\theta_zC\theta_x \\ 0. & C\theta_yC\theta_x & -C\theta_yS\theta_x \end{bmatrix} \qquad (4.13.a)$$

$$\partial RPY(\theta)/\partial\theta_y = \begin{bmatrix} -C\theta_zS\theta_y & C\theta_zC\theta_yS\theta_x & C\theta_zC\theta_yC\theta_x \\ -S\theta_zS\theta_y & S\theta_zC\theta_yS\theta_x & S\theta_zC\theta_yC\theta_x \\ -C\theta_y & S\theta_yS\theta_x & S\theta_yC\theta_x \end{bmatrix} \qquad (4.13.b)$$

$$\partial RPY(\theta)/\partial\theta_z = \begin{bmatrix} -S\theta_zC\theta_y & -S\theta_zS\theta_yS\theta_x-C\theta_zC\theta_x & -S\theta_zS\theta_yC\theta_x+C\theta_zS\theta_x \\ C\theta_zC\theta_y & C\theta_zS\theta_yS\theta_x-S\theta_zC\theta_x & C\theta_zS\theta_yC\theta_x+S\theta_zS\theta_x \\ 0. & 0. & 0. \end{bmatrix} \qquad (4.13.c)$$

where $C\theta_x = \cos(\theta_x)$, $S\theta_x = \sin(\theta_x)$, $C\theta_y = \cos(\theta_y)$, $S\theta_y = \sin(\theta_y)$, $C\theta_z = \cos(\theta_z)$, and $S\theta_z = \sin(\theta_z)$.

With (4.7) and (4.13), we have:

$$\partial O(k+1|k)/\partial\theta_x(k|k) = (\partial RPY(\theta(k|k))/\partial\theta_x(k|k))\ RPY(\Delta\theta(k|k)) \qquad (4.14.a)$$

$$\partial O(k+1|k)/\partial\theta_y(k|k) = (\partial RPY(\theta(k|k))/\partial\theta_y(k|k))\ RPY(\Delta\theta(k|k)) \qquad (4.14.b)$$

$$\partial O(k+1|k)/\partial\theta_z(k|k) = (\partial RPY(\theta(k|k))/\partial\theta_x(k|k))\ RPY(\Delta\theta(k|k)) \qquad (4.14.c)$$

$$\partial O(k+1|k)/\partial\Delta\theta_x(k|k) = RPY(\theta(k|k))\ (\partial RPY(\Delta\theta(k|k))/\partial\Delta\theta_x(k|k)) \qquad (4.14.d)$$

$$\partial O(k+1|k)/\partial\Delta\theta_y(k|k) = RPY(\theta(k|k))\ (\partial RPY(\Delta\theta(k|k))/\partial\Delta\theta_y(k|k)) \qquad (4.14.e)$$

$$\partial O(k+1|k)/\partial\Delta\theta_z(k|k) = RPY(\theta(k|k))\ (\partial RPY(\Delta\theta(k|k))/\partial\Delta\theta_z(k|k)) \qquad (4.14.f)$$

Since O(k+1) is independent of p(k) and Δp(k), it is obvious that

$$\partial O(k+1|k)/\partial \mathbf{p}^T(k|k) = 0 \text{ and } \partial O(k+1|k)/\partial \Delta \mathbf{p}^T(k|k) = 0. \tag{4.15}$$

With (4.14) and (4.15), we can now compute $\partial\theta(k+1|k)/\partial \mathbf{b}^T(k|k)$. Taking derivatives of (4.10), (4.11), and (4.12) with respect to b(k), we get:

$$\begin{aligned}
&\partial\theta x(k+1|k)/\partial \mathbf{b}^T(k|k) = \partial \tan^{-1}(o_{32}(k+1|k)/o_{33}(k+1|k))/\partial \mathbf{b}^T(k|k) \\
&= [(\partial o_{32}(k+1|k)/\partial \mathbf{b}^T(k|k))o_{33}(k+1|k) - o_{32}(k+1|k)(\partial o_{33}(k+1|k)/\partial \mathbf{b}^T(k|k))] \\
&\quad [o_{32}^2(k+1|k) + o_{33}^2(k+1|k)]^{-1}
\end{aligned} \tag{4.16.a}$$

$$\begin{aligned}
&\partial\theta y(k+1|k)/\partial \mathbf{b}^T(k|k) \\
&= \partial \tan^{-1}(-o_{31}(k+1|k)/(o_{21}^2(k+1|k) + o_{11}^2(k+1|k))^{1/2})/\partial \mathbf{b}^T(k|k) \\
&= [-\partial o_{31}(k+1|k)/\partial \mathbf{b}^T(k|k)]\ [o_{21}^2(k+1|k) + o_{11}^2(k+1|k)]^{1/2} \\
&+ [o_{11}(k+1|k)(\partial o_{11}(k+1|k)/\partial \mathbf{b}^T(k|k)) + o_{21}(k+1|k)(\partial o_{21}(k+1|k)/\partial \mathbf{b}^T(k|k))] \\
&\quad o_{31}(k+1|k)[o_{21}^2(k+1|k) + o_{11}^2(k+1|k)]^{-1/2}
\end{aligned} \tag{4.16.b}$$

$$\begin{aligned}
&\partial\theta z(k+1|k)/\partial \mathbf{b}^T(k|k) = \partial \tan^{-1}(o_{21}(k+1|k)/o_{11}(k+1|k))/\partial \mathbf{b}^T(k|k) \\
&= [(\partial o_{21}(k+1|k)/\partial \mathbf{b}^T(k|k))o_{11}(k+1|k) - o_{21}(k+1|k)(\partial o_{11}(k+1|k)/\partial \mathbf{b}^T(k|k))] \\
&\quad [o_{21}^2(k+1|k) + o_{11}^2(k+1|k)]^{-1}
\end{aligned} \tag{4.16.c}$$

By using (4.13)-(4.16), we have the 12x12 matrix:

$$\partial(f(\mathbf{b}(k|k))/\partial \mathbf{b}^T(k|k) = \partial(\mathbf{b}(k+1|k)/\partial \mathbf{b}^T(k|k)$$

$$= \begin{bmatrix} \partial\theta(k+1|k)/\partial\theta(k|k) & 0 & \partial\theta(k+1|k)/\partial\Delta\theta(k|k) & 0 \\ 0 & I & 0 & I \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \end{bmatrix}$$

where each element is an 3x3 submatrix, and 0 and I are a 3x3 zero matrix and a 3x3 identity matrix, respectively.

4.3 The Measurement Model

For each kind of image primitive, there is a relation between a primitive's image attributes in $\mathbf{d}(k+1)$ and the unknown parameters in $\mathbf{b}(k+1)$. Thus, it is convenient to discuss each type of primitive individually. we will discuss the point primitive first, then the line primitive. and, finally, the elliptical/hyperbolical primitive. In our discussion below, the index i is for the i-th correspondence.

<u>The Point Primitives</u>. Let $\mathbf{r}_i$ be the point in the model and (X_i, Y_i) be the corresponding point in the image. It can be shown from (1.1) and (1.2) that, under perspective projection, we have the pair of equations:

$$d_{i,1}(k+1) = [f,\ 0,\ X_i(k+1)]\ (O(k+1)\ \mathbf{r}_i + \mathbf{p}(k+1)) = 0. \qquad (4.17.a)$$

$$d_{i,2}(k+1) = [0,\ f,\ Y_i(k+1)]\ (O(k+1)\ \mathbf{r}_i + \mathbf{p}(k+1)) = 0. \qquad (4.17.b)$$

where f is the focal length, and $O(k+1) = RPY(\theta(k+1))$.

In the above equation, the elements of $\mathbf{d}_i(k+1)$ do not represent physical attributes, but they are linear combinations of physical attributes. This particular choice of $\mathbf{d}_i(k+1)$ yields a relatively simple vector function $\mathbf{h}(*)$. This is an advantage because the simpler $\mathbf{h}(*)$ in (4.2) the simpler will be. the behavior of the inner loop of the *DynamicEKF* algorithm in Appendix E. Since the inner loop is essentially a nonlinear optimization process, a simple $\mathbf{h}(*)$ will be helpful in avoiding the undesirable local minimums. The formulation in (4.17) always makes the measured attributes in $\mathbf{d}_i(k+1)$ zero. In contrast, the extrapolated attributes are given by:

$$d_{i,1}(k+1|k) = [f,\ 0,\ X_i(k+1)]\,(O(k+1|k)\ \mathbf{r}_i + \mathbf{p}(k+1|k)) \qquad (4.18.a)$$

$$d_{i,2}(k+1|k) = [0,\ f,\ Y_i(k+1)]\,(O(k+1|k)\ \mathbf{r}_i + \mathbf{p}(k+1|k)) \qquad (4.18.b)$$

It is worth mentioning that $O(k+1|k)$ and $p(k+1|k)$ in (4.18) can be obtained before $X_i(k+1)$ and $Y_i(k+1)$ are actually measured, but $d_i(k+1|k)$ cannot be obtained prior to measurement at time k+1. However, the inability to calculate $d_i(k+1|k)$ does not hinder the application of prediction (e.g., the use of prediction to establish correspondences), for $O(k+1|k)$, $p(k+1|k)$, $X_i(k+1|k)$ and $Y_i(k+1|k)$ are the predictions that are needed, not $d_i(k+1|k)$ in (4.18). As mentioned earlier, $O(k+1|k)$ and $p(k+1|k)$ can be obtained by using the dynamic model derived in the previous section. On the other hand, to calculate $X_i(k+1|k)$ and $Y_i(k+1|k)$, we have to use the following pair of equations:

$$X_i(k+1|k) = f\, x_i^c(k+1|k) \;/\; z_i^c(k+1 \tag{4.19.a}$$

$$Y_i(k+1|k) = f\, y_i^c(k+1|k) \;/\; z_i^c(k+1|k) \tag{4.19.b}$$

where $r_i^c(k+1|k)=[x_i^c(k+1|k),y_i^c(k+1|k),z_i^c(k+1|k)]^T = O(k+1|k)r_i+p(k+1|k)$.

For a point primitive, the derivative of $d_i(k+1|k)$ with respect to $b(k+1|k)$ is given by:

$$\partial \mathbf{d}_i(k+1|k)/\partial b^T(k+1|k)=\begin{bmatrix} f, & 0, & X_i(k+1) \\ 0, & f, & Y_i(k+1) \end{bmatrix}\begin{bmatrix} & & & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \mathbf{w}_x & \mathbf{w}_y & \mathbf{w}_z & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & & & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

where $\mathbf{w}_x = \partial O(k+1|k)/\partial\theta_x(k+1|k))\mathbf{r}_i$, $\mathbf{w}_y = \partial O(k+1|k)/\partial\theta_y(k+1|k))\mathbf{r}_i$, and $\mathbf{w}_z = (\partial O(k+1|k)/\partial\theta_z(k+1|k))\mathbf{r}_i$. The first matrix on the right hand side of the previous equation is 2x3, and the second is 3x12; thus, $\partial \mathbf{d}_i(k+1|k)/\partial b^T(k+1|k)$ is a 2x12 matrix.

<u>The Line Primitives</u>. Let $\mathbf{a}_i$ be the unit directional vector of the model line and $\mathbf{r}_i$ be any point on the model line. Again, let the corresponding image line be described by the slope m_i and the intercept T_i. It is shown in Appendix D.2 that, under perspective projection, we have the pair of equations:

$$d_{i1}(k+1) = \mathbf{q}_i^T(k+1)\ O(k+1)\ \mathbf{a}_i = 0. \qquad (4.20.a)$$

$$d_{i2}(k+1) = \mathbf{q}_i^T(k+1)\ (O(k+1)\ \mathbf{r}_i + p(k+1)) = 0. \qquad (4.20.b)$$

where f is the focal length, and $\mathbf{q}_i(k+1) = [-f\ m_i(k+1),\ f,\ T_i(k+1)]^T$.

Again, the attributes used in (4.20) are linear combinations of the physical attributes m_i and T_i. Also, $\mathbf{d}_i(k+1)$ is always equal to zero, and the calculation of $\mathbf{d}_i(k+1|k)$ is given by:

$$d_{i1}(k+1|k) = q_i^T(k+1)\ O(k+1|k)\ \mathbf{a}_i \qquad (4.21.a)$$

$$d_{i2}(k+1|k) = q_i^T(k+1)\ (O(k+1|k)\ \mathbf{r}_i + p(k+1|k)) \qquad (4.21.b)$$

For a line primitive, the derivative of $\mathbf{d}_i(k+1|k)$ with respect to $\mathbf{b}(k+1|k)$ is given by:

$$\partial d_{i1}(k+1|k)/\partial \mathbf{b}^T(k+1|k) = q_i^T(k+1) \begin{bmatrix} & & & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \mathbf{cx} & \mathbf{cy} & \mathbf{cz} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & & & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\partial d_{i2}(k+1|k)/\partial \mathbf{b}^T(k+1|k) = q_i^T(k+1) \begin{bmatrix} & & & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \mathbf{wx} & \mathbf{wy} & \mathbf{wz} & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & & & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

where $\mathbf{cx} = \partial O(k+1|k)/\partial\theta x(k+1|k))\mathbf{a}_i$, $\mathbf{cy} = \partial O(k+1|k)/\partial\theta y(k+1|k))\mathbf{a}_i$, $\mathbf{cz} = (\partial O(k+1|k)/\partial\theta z(k+1|k))\mathbf{a}_i$, $\mathbf{wx} = \partial O(k+1|k)/\partial\theta x(k+1|k))\mathbf{r}_i$, $\mathbf{wy} = (\partial O(k+1|k)/\partial\theta y(k+1|k))\mathbf{r}_i$, and $\mathbf{wz} = (\partial O(k+1|k)/\partial\theta z(k+1|k))\mathbf{r}_i$.

The Elliptic Primitives. Since the derivation of elliptic and hyperbolic primitives are quite similar, here we only consider the elliptic primitive. Let the transformation of the primitive-coordinate basis to the object coordinate basis be described by the matrix Q_i and the vector s_i, and the transformation of the object to the camera be described by the matrix O(k+1) and the vector p(k+1). The transformation between the primitive and the camera coordinate bases is then given by $G_i(k+1) = O(k+1)Q_i$ and $\mathbf{t}_i(k+1) = O(k+1)s_i + p(k+1)$. Let the model ellipse be the intersection of $x^2/\alpha^2 + y^2/\beta^2 = 1$ and $z = 0$, where x, y, and z are in primitive coordinate basis. Also, let $\mathbf{w} =$

$[X,Y,1]^T$ and $\mathbf{u}_i(k+1) = G_i^T(k+1)t_i(k+1)$. Following Appendix A.1, we have the elliptic equation:

$$\mathbf{w}^T A_i(k+1)\mathbf{w} = \mathbf{w}^T\, D\, G_i(k+1)\, B_i(k+1)\, G_i^T(k+1)\, D^T\, \mathbf{w} = 0 \qquad (4.21)$$

where $B(k) = \begin{bmatrix} u_{i,3}^2(k)/\alpha^2 & 0 & -u_{i,1}(k)u_{i,3}(k)/\alpha^2 \\ 0 & u_{i,3}^2(k)/\beta^2 & -u_{i,2}(k)u_{i,3}(k)/\beta^2 \\ -u_{i,1}(k)u_{i,3}(k)/\alpha^2 & -u_{i,2}(k)u_{i,3}(k)/\beta^2 & u_{i,1}^2(k)/\alpha^2 + u_{i,2}^2(k)/\beta^2 - 1 \end{bmatrix}$,

$D = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -f \end{bmatrix}$, and $u_{i,j}(k)$ is the j-th element of the vector $\mathbf{u}_i(k)$.

Let the image ellipse be described by $\mathbf{w}^T M_i(k+1)\mathbf{w} = 0$, where the (3,3) element in $M_i(k+1)$ is set to one. By comparing elements in $M_i(k+1)$ and $A_i(k+1)$, we have

$$d_{i,2j+n-2}(k+1) = M_{i,jn}(k+1)\,A_{i,33}(k+1) - A_{i,jn}(k+1) = 0 \qquad (4.22)$$

for every element (j,n) in the upper triangle submatrices of $M_i(k+1)$ and $A_i(k+1)$, except the element (3,3).

Consequently, $d_i(k+1|k)$ is given by

$$d_{i,2j+n-2}(k+1|k) = M_{i,jn}(k+1)\;A_{i,33}(k+1|k) - A_{i,jn}(k+1|k) \qquad (4.23)$$

for every element (j,n) in the upper triangle submatrices of $M_i(k+1)$ and $A_i(k+1|k)$, except the element (3,3).

For an elliptic primitive, the derivative of $d_i(k+1|k)$ with respect to $b(k+1|k)$ is given by

$$\partial d_{i,2j+n-2}(k+1|k)/\partial b^T(k+1|k) = M_{i,jn}(k+1)\;(\partial A_{i,33}(k+1|k)/\partial b^T(k+1|k)) \\ - (\partial A_{i,jn}(k+1|k)/\partial b^T(k+1|k)) \qquad (4.24)$$

for every element (j,n) in the upper triangle submatrices of $M_i(k+1)$ and $A_i(k+1|k)$, except the element (3,3).

Chapter Summary. Although the 3D object motion between two sample instants can be extracted by working out the difference between the object's locations at those two instants (object location at each instant can be determined by using algorithms in the previous chapter), we have presented an Extended Kalman Filter approach, in this chapter, to recover smoothed 3D motion from a sequence of images. In EKF formulation, dynamics and measurement models are built to extrapolate an estimate of unknown parameters (including 6 location parameters and 6 motion parameters) and to produce a minimum uncertainty estimate from the extrapolated estimate and measured image primitive attributes. The image primitives are ellipse, hyperbola, line, and point. The recovered 3D motion of an object can be used for tracking the object, grasp formation, and visual feedback. In the next chapter, implem[illegible] the techniques presented in this and previous cha[illegible]ussed.

Chapter 5. Implementations

This chapter shows a camera calibration scheme (Section 5.1), a partial implementation of the proposed "locating-labeling" method for object recognition and localization (Section 5.2), and a partial implementation of the proposed motion recovery method (Section 5.3). In this thesis, the focal length associated with each image taken for recognition, localization, and motion recovery is assumed to be known. The known focal length is the result of camera calibration. The 3D object recognition subsystem is implemented with only line and point primitives; thus only polyhedral objects are recognizable in the present implementation. In motion recovery experiment, the object considered is moved by a moving device independent of the vision system. Currently, only point correspondences are used for recovering the motion. The motion recovery subsystem was completed one year before the object recognition subsystem. At this moment, the recognition and motion recovery subsystems are are not intergrated.

5.1 Camera Calibration Subsystem

In this thesis, the focal length of the camera is assumed to be known. The assumption comes from the fact that a table of different focal length values for different positions of the focusing lens of the camera can be computed off line. When an image is taken, the focal length associated with the lens position is retrieved by the proposed system from the table. In the case of an auto-focus camera, the positions of the focusing lens could be obtained from the revolutions of the driving motor.

The camera calibration is to establish the table of focal lengths for different lens positions. Currently, a procedure from Tsai (1986) is used to perform the calibration. The procedure is slightly modified to suit our application and is summarized in Appendix F. The camera calibrating procedure in Appendix F estimates the transformation between a reference coordinate basis and the camera coordinate basis, as well as certain camera attributes (focal length and distortion factors). The input data to the procedure are a set of coordinates of points in a reference coordinate basis and their identified counterparts in the digitized image. At least 7 correspondences between noncoplanar points in the reference space and points in the images are needed for this algorithm to work. More pairs of points will, in general, give better results.

The noncoplanar points in the reference coordinate basis can be produced by moving a plate of points to different positions or by using a cube box with points on two of its faces. Both are implemented, but only the former will be reported here with experimental data.

The plate used contains 42 points. Their positions relative to the plate coordinate basis defined on the plate is shown in Figure 5.1. The plate is moved by a motion generating device to different positions, thus resulting in noncoplanar points in a fixed reference coordinate basis. Images are taken for the plate at different positions. In taking a set of images, the camera can be arbitrarily placed at any position and viewing direction as long as points in the reference space can be viewed by the camera.

The motion-generating device has a clamp-vice for holding an object which, in this case, is the calibration plate. A drawing of the configuration of the device, the plate and the camera is shown in Figure 5.2 and a photograph is shown in Figure 5.3.(a). The moving device is driven by two separate stepping motors, one for translation and the other for rotation. A close look of the device is given in Figure 5.3.(b). The translation axis can be placed at any direction relative to the camera to generate general 3D translation, and the rotation axis can be fixed at any direction of the upper sphere to produce general 3D rotation. The direction of the rotation axis is adjusted by manually tuning the first two Euler angles, as shown in Figure 5.3.(b). The angle of rotation about the rotation axis is controlled by the second stepping motor, which is placed at the axis of the third Euler angle.

Let the reference coordinate basis, e^r, be at the far end of the translational track of the moving device in Figure 5.3.(a) and the translation is along the x^r axis. By programming the moving device to move the plate to four different positions, $x^r = 30.$, $x^r = 40.$, $x^r = 50.$, and $x^r = 60$ with $y^r = 0$, $z^r = 0$, and no rotation, we digitized four images in the left column of Figure 5.4. The coordinates of the points on the plate relative to e^r can be calculated by simply adding the coordinates in e^p and the translation between e^p and e^r.

To measure X and Y coordinates of the image points, each image is first passed through a Sobel operator to extract edges. The extracted edge images are shown in the right column of Figure 5.4. The plate in each edge image is identified by looking for the pattern of a four line quadrilateral. Points in the quadrilateral are detected, and their coordinates in the image are measured. The X and Y coordinates of each point in the edge image is measured by averaging the X and Y coordinates of pixels representing the image point.

The measured X and Y coordinates of image points and the known coordinates of their counterparts in the reference space are used by the calibrating procedure to recover the unkown attributes for the camera and the unknown transformation of the reference coordinate basis to the camera basis.

With the four images in Figure 5.4, we have 168 (4x42) sets of coordinates of points in e^r and X, Y coordinates of their corresponding image points in the image for the calibrating algorithm. The results of this calibration run is summarized as follows:

$$G = \begin{bmatrix} 0.457226 & 0.888811 & 0.030977 \\ 0.013799 & 0.021095 & -0.999682 \\ -0.889182 & 0.457508 & -0.002619 \end{bmatrix}$$

$$t = [-22.425341 \quad 11.557964 \quad -144.681708]^T$$

$$f_m = 1206.259235, \quad s = 0.797256$$

$$k_{1m} = -1.725762e-7, \quad k_{2m} = 7.867127e-13$$

where G and t define the transformation of the reference coordinate basis to the camera coordinate basis. f_m is the focal length for image coordinates in pixels, s is the ratio between vertical scale and horizontal scale in the image, and k_{1m} and k_{2m} are the second-order and fourth-order radial lens distortion factors. More precise definition of these parameters is given in Appendix F.

The calibrated camera model has error measures:

Average square error = 0.385524 pixels

Maximum square error = 1.157407 pixels

where Average square error $= \sum_{i=1}^{168} [(X_{mi} - \hat{X}_{mi})^2 + (Y_{mi} - \hat{Y}_{mi})^2]^{1/2} / 168$;

Maximum square error $= \operatorname{Max}_{i=1}^{168} [(X_{mi} - \hat{X}_{mi})^2 + (Y_{mi} - \hat{Y}_{mi})^2]^{1/2}$;

X_{mi} and Y_{mi} are the measured X and Y coordinates of each image point; and $\hat{X}_{mi}$ and $\hat{Y}_{mi}$ are the extrapolated X and Y coordinates in the image for each corresponding point in e^r by using the estimated camera parameters.

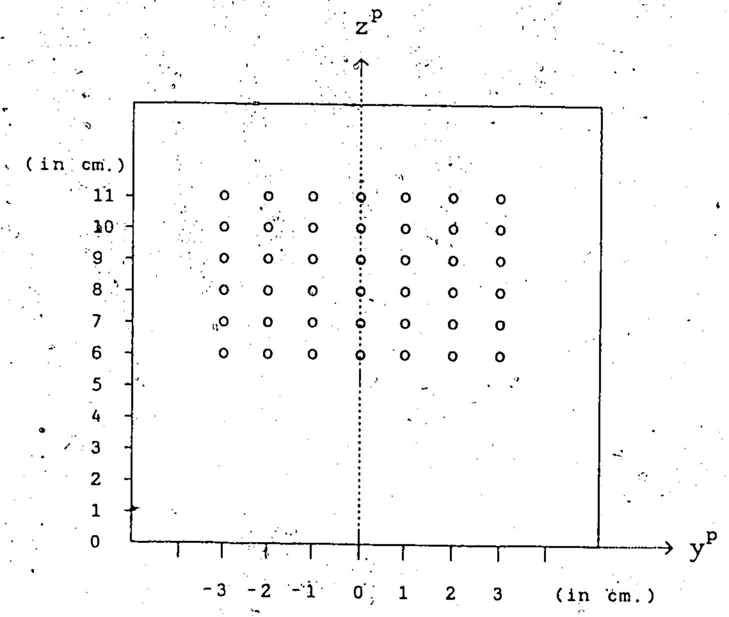


Figure 5.1 The calibration points on the calibration plate. The solid lines define the boundary of the plate, and dashed lines are for the axes of the plate coordinate basis. The plate is defined on the plane, $x^p=0$. Two adjacent points in the same row or column are one centimeter apart. The origin of the plate coordinate basis is at the center of the bottom border of the plate.

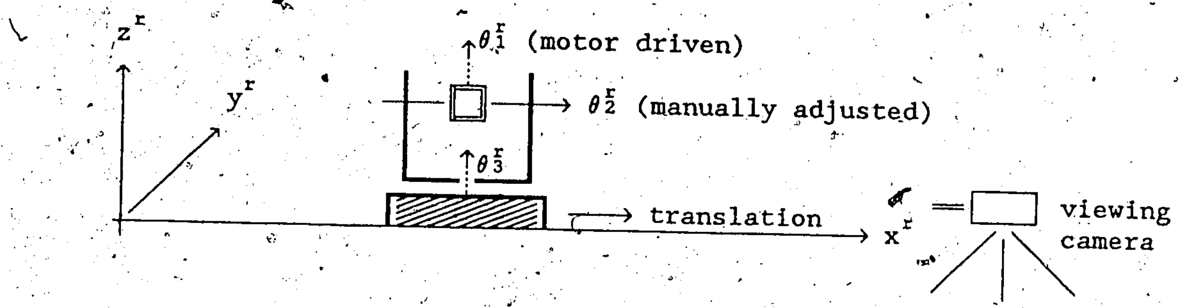
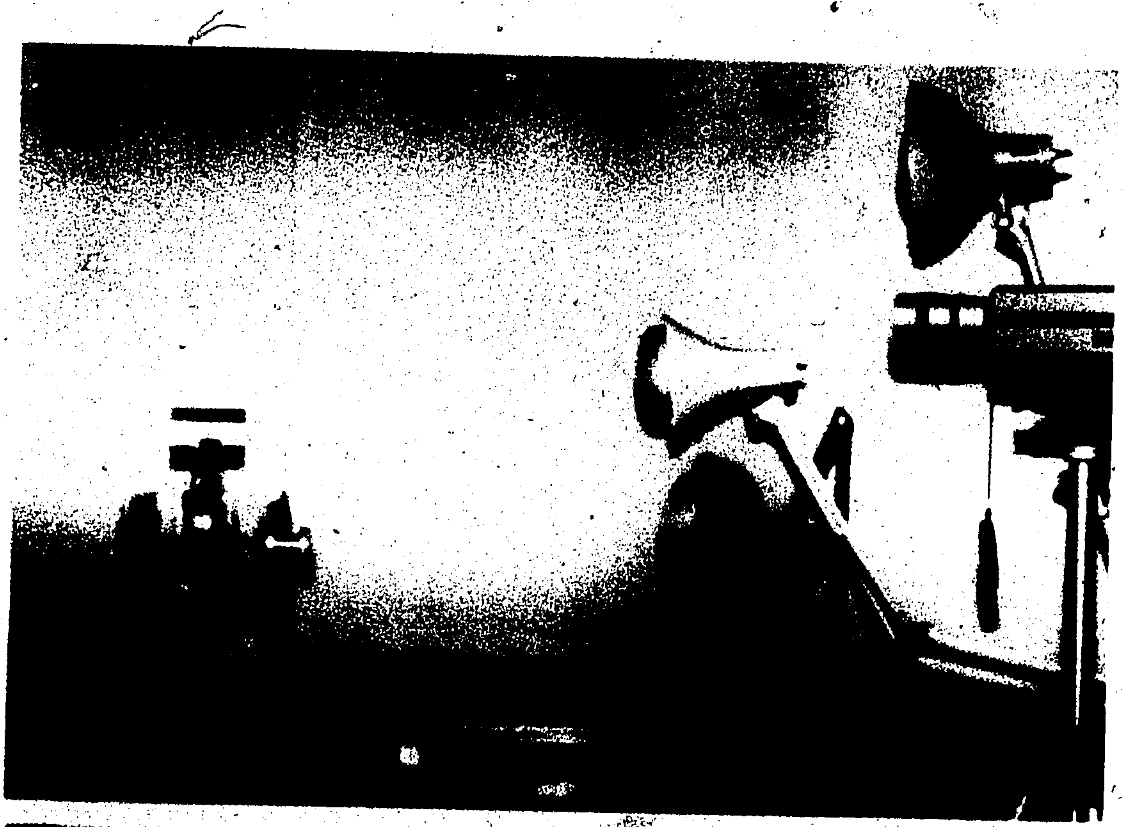


Figure 5.2 The reference coordinate basis defined in the motion-generating device. The moving part is driven by a stepping motor and translated along x^r axis. The rotation is defined in Euler angles, in which a general rotation is decomposed into the sequence of three rotations around the reference coordinate axes. The three rotations in sequence are Rot(z^r,θ_1^r), Rot(x^r,θ_2^r), Rot(z^r,θ_3^r), as shown in dashed arrows. The rotation θ_1^r is driven by a second stepping motor, while θ_2^r and θ_3^r are adjusted manually to set the direction of a general rotation axis. The camera on a tripod can be placed at any direction and position relative to the reference coordinate basis as long as the object held by the moving part is in the camera's field of view.

Figure 5.3. One view of the motion generating device. (a) The configuration of themoving device, the calibration plate, and the camera. (b) A closelook of the moving part of the device.



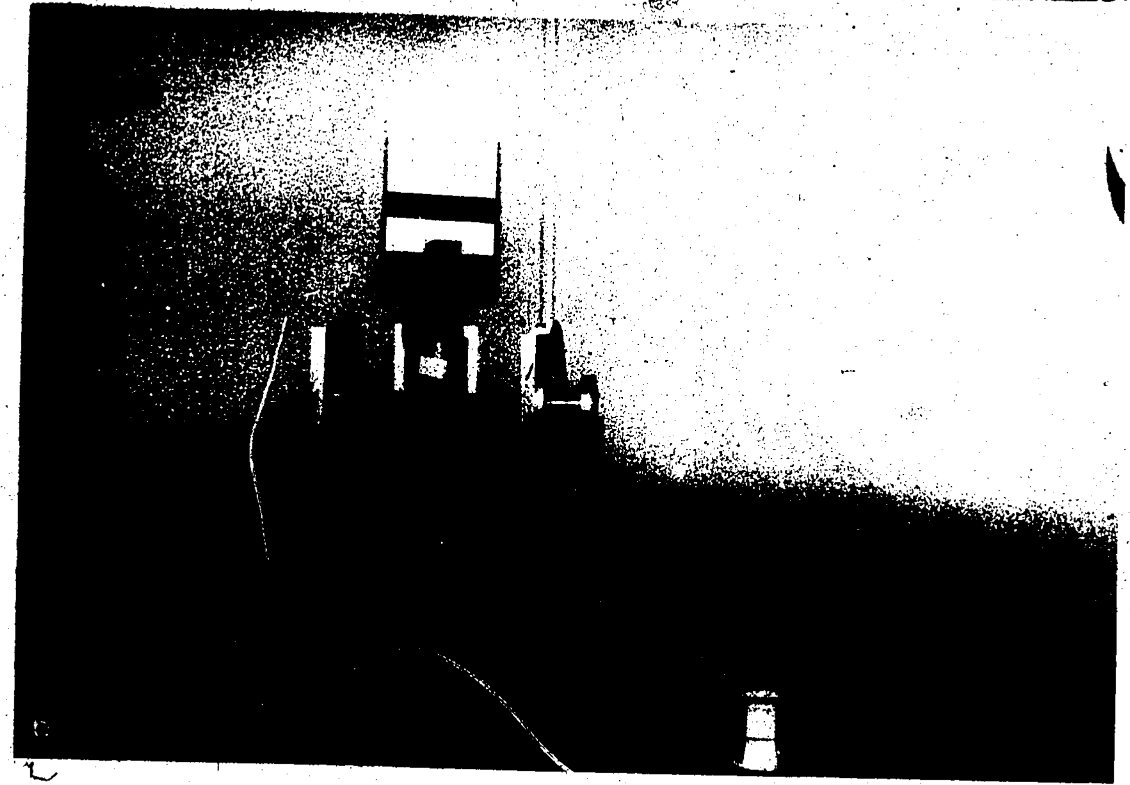
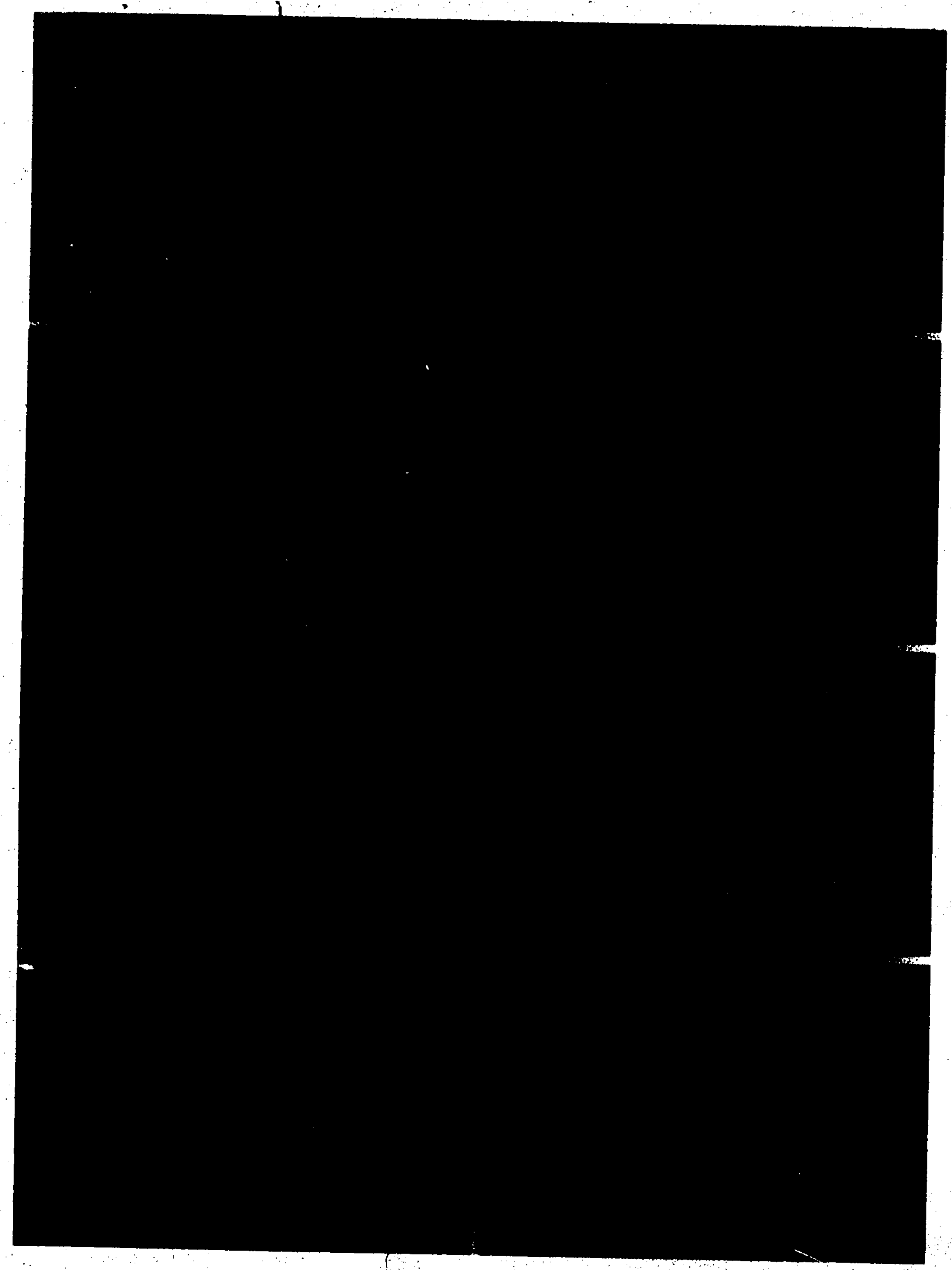


Figure 5.4 A sequence of images used in camera calibration. The left column contains digitized intensity images, and the right column contains processed edge images.



5.2 Object Recognition and Localization Subsystem

In this section, the techniques described in Chapters 2 and 3 are used to implement an object recognition and localization subsystem. The present implementation is limited to line and point primitives. In the following experiment, only one object model is used; the object is an audio-cassette tape. The wireframe model of the cassette is shown in Figure 5.5, and the object coordinate basis is shown in Figure 5.6. The primitives used are lines and points, and are stored in the model database. In addition, the relations between lines, such as parallel, intersection, and coplanar, are stored symbolically. The surfaces of the object are also stored for testing occlusion.

An image of the cassette is shown in Figure 5.7.(a). The image is taken with $f_m = 2312.734174$ and $s = 0.855635$ (The ratio of vertical scale to horizontal scale is different from the ratio, s, in Section 5.1, because a different camera, though of the same type of RCA camera, was used for this experiment. The ratio, s, in general remains the same for a given camera, regardless of the change of focal length).

The intensity image in Figure 5.7.(a) is passed through a Sobel operator to extract the edges for pixels with intensity changes. The edge image is shown in Figure 5.7.(b). The edges are further thinned to single pixel width, as shown in Figure 5.7.(c), and the thinned edges are passed through the two-stage primitive detector (discussed in Chapter 2) to extract conic segments, line segments, and critical points. The detected primitives are stored with the cell structure introduced in Chapter 2. Currently, only line cells are used for recognition. The line cells are accessed through a line table, a slope list, or a pixel list. The line table uses endpoints of image lines as keys to store pointers to line cells. The slope list and pixel list contain pointers to line cells. The pointers in the slope list are sorted according to the slope of each line, and the pointers in the pixel list are sorted with the number of pixels in each line.

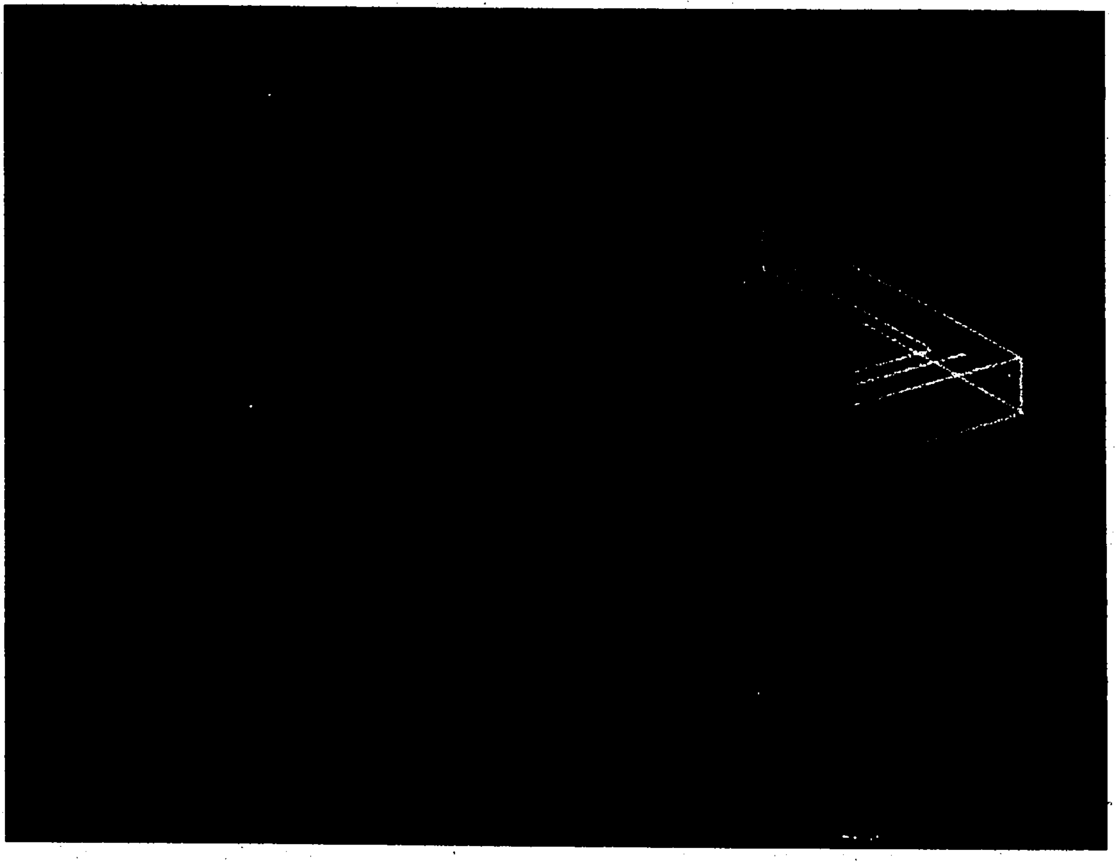
The organized image primitives are grouped into parallel lines, parallel lines with another line intersecting them, U-shape of three lines, two pairs of parallel lines forming a quadrilateral, and a quadrilateral of four lines. These groups are used as index feature groups to formulate minimum-correspondence hypotheses.

Each hypothesis is tested by the hypothesis tester introduced in Section 2.4. The testing of the hypothesis is done by cycling the four processes; locating the object by using the hypothesized matches, testing consistency of the matches with the recovered object location, predicting other image primitives by using the recovered object location and the object model, and augmenting the set of hypothesized matches between model primitives and image primitives. Figure 5.8 shows a set of hypothesized matches being augmented and tested. At first, three line matches are used to form a minimum-correspondence hypothesis. The matches are tested to be consistent with the recovered object location, as shown in Figure 5.8.(a). Subsequently, another line correspondence is hypothesized and added to the existing three. The augmented set of hypothesized matches again are tested to be consistent with the recovered object location, as shown in Figure 5.8.(b). The localization, consistency testing, prediction, and augmentation continue until the set of hypothesized matches meet the termination criteria in the initiated model; in this case, 8 consistent line matches.

After recognition is confirmed, the object location is recovered again with all consistent matches. For the image in Figure 5.7.(a), the recovered roll, pitch, and yaw angles, in radians, of the cassette relative to the camera are 0.083818, 0.177362, 0.460718, respectively. The position, in centimeters, of the cassette relative to the camera is recovered as the vector $[-0.198683\ -1.670742\ -98.425624]^T$. The model primitives of the cassette are projected with the recovered location and superimposed with the thinned edges in Figure 5.7.(c). The superimposition is shown in Figure 5.7.(d).

Several trials were run with this implementation on a IBM AT with a 80287 math coprocesser. In most cases, the hypothesis formulation and testing process takes less than 60 secs, which includes the superimposition of every set of hypothesized matches with the edge image on a monitor driven by an ITI (Imaging Technology Inc.) Series 100 Image Processing Board. In this implementation, the most time consuming part was the edge extracting and thinning processes. It can take up to 3 minutes, because of the inherent sequential nature of this implementation. However, this can be overcome by employing a commercially available high-speed image processor with parallel convolvers and pipeline array processors.

Figure 5.5 One view of the wireframe model of the audio cassette.



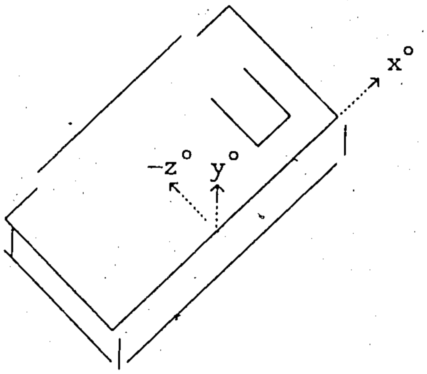


Figure 5.6 The object coordinate basis defined in the audio cassette.

Figure 5.7 The images used in the recognition process. (a) A single intensity image. (b) The edge image extracted with the Sobel operator. (c) The thinned edge image. (d) The superimposition of model wireframe projected using recovered location with the thinned edge image. Each projected line is tested against surfaces in the model to see if they are occluded. If so, that line is removed.

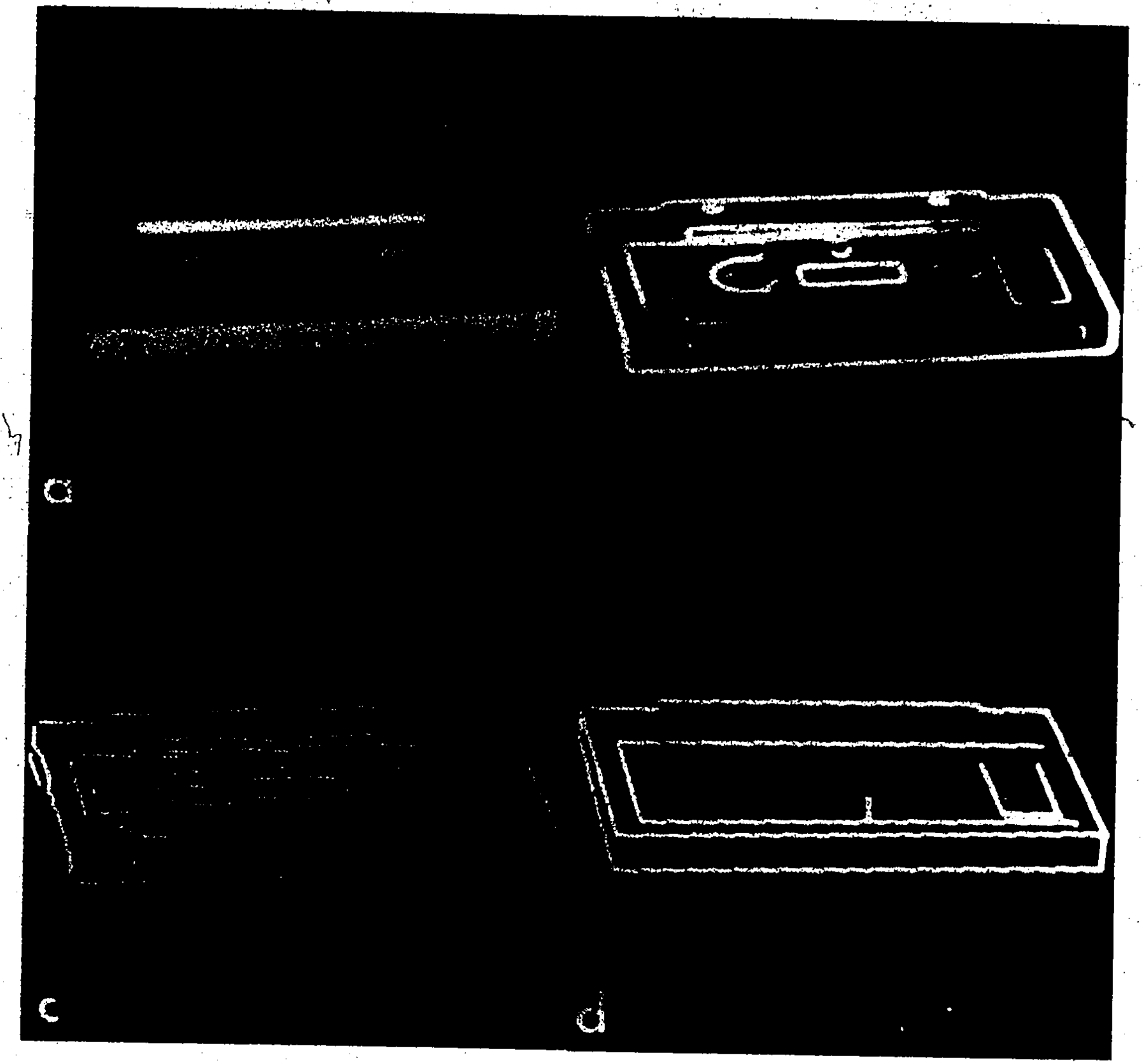
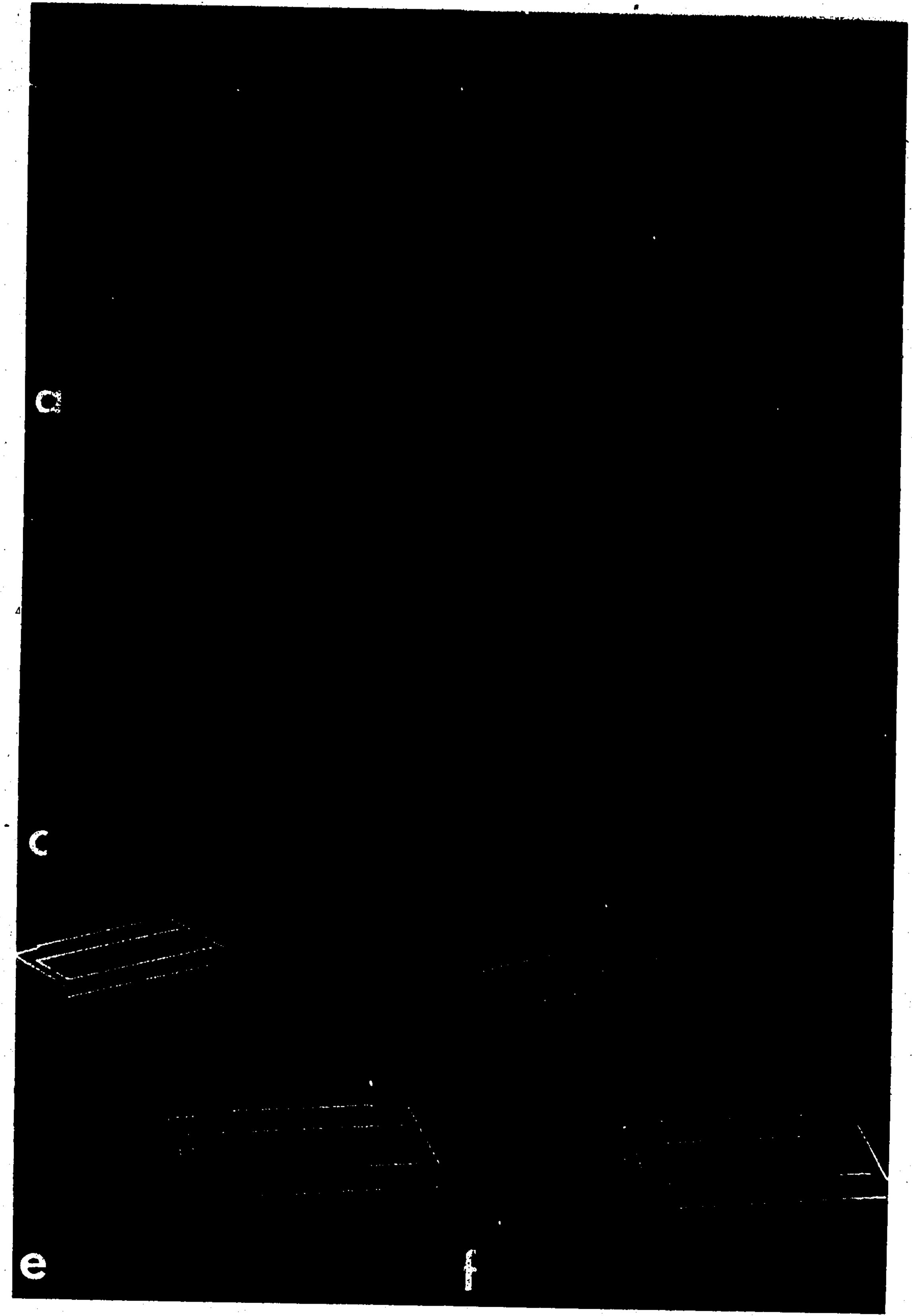

a
c
d

Figure 5.8 The hypothesis testing cycle of localization, consistency testing, prediction, and augmentation. The hypothetically matched model lines are projected using recovered object location and superimposed with detected image lines. Consistency testing is conducted by comparing the error of the superposition. When tested to be consistent with recovered object location, hypothesized matches are augmented by including the most significant predicted match. The following contains the snapshots of the recognition process (a) three hypothesized matches. After location is estimated from the matches, the matched model lines are projected (as the bright lines) and are compared with detected image lines (the dimmed lines). It can be seen that if the three hypothesized matches are all valid, the projected object almost coincides the detected one on the image. (b) Four hypothesized matches with one additional match to the three consistent matches in (a). (c) Five hypothesized matches. (d) Six hypothesized matches. (f) Seven hypothesized matches. (e) Eight hypothesized matches. At this stage, it can be seen that the projected object coincides with the detected one on the image.


a
c
e
f

5.3 Motion Recovery Subsystem

The motion recoverer subsystem here is tailored for the estimation of the motion of object recognized by the proposed recognition scheme, which was outlined in Chapter 2 and implemented in Section 5.2. The recovered motion of a object can be used for grasping the object as well as aiding object recognition. In this experiment, we assumed the object had already been recognized, and hence proceeded to recover its motion by using point correspondences between the image and the model. The object used here is the calibration plate shown in Figure 5.1., and the motion is generated by the device shown in Figure 5.2.

By programming the device to start at $x^r = 30.$ and θ_z^r (or θ_1^r in Figure 5.2) = -0.698132 radians (or -40. degrees), and the incremental motion between two image frames as $\Delta x^r = 10.$ cm. and $\Delta\theta_z^r = 0.15708$ radians (or 9. degrees), we have the trajectory in Table 5.1 for six image frames (the choice of incremental rotation as 9 degrees per frame is based on the fact that the second stepping motor's resolution is 0.9 degrees per step).

Table 5.1 The trajectory of the programmed motion relative to the reference coordinate basis at the far end, from the camera, of the motion-generating device in Figure 5.2.

Frame number k	$\theta_z^r(k)$	$p_x^r(k)$	$\Delta\theta_z^r(k)$	$\Delta p_x^r(k)$
1	-0.698132	30	0.15708	10
2	-0.541052	40	0.15708	10
3	-0.383972	50	0.15708	10
4	-0.226893	60	0.15708	10
5	-0.069813	70	0.15708	10
6	0.087266	80	0.15708	10

Note. Angles are in radians, and distances are in centimeters. $\Delta p_x^r(k) \equiv p_x^r(k) - p_x^r(k-1)$, and $\Delta\theta_z^r(k) \equiv \theta_z^r(k) - \theta_z^r(k-1)$. Other variables, $\theta_x^r(k)$, $\theta_y^r(k)$, $p_y^r(k)$, and $p_z^r(k)$, are all zero. Here $\theta_z^r(k)$ is in roll-pitch-yaw expression, which is equivalent to $\theta_1^r(k)$ in Figure 5.2, in this case.

Though the motion in Table 5.1 is 1D rotation and 1D translation relative to the reference coordinate basis, it is 3D rotation and 3D translation after conversion to the camera coordinate basis. Hence, it is a 3D motion as far as the camera is concerned. In this experiment, the transformation of the reference coordinate basis to the camera coordinate basis is the same as one found by the camera calibration in Section 5.1. The transformation is described by:

$$G = \begin{bmatrix} 0.457226 & 0.888811 & 0.030977 \\ 0.013799 & 0.021095 & -0.999682 \\ -0.889182 & 0.457508 & 0.002619 \end{bmatrix},$$

and

$$t = [-22.425341 \quad 11.557964 \quad -144.681708]^T$$

Other camera attributes from the calibration in Section 5.1 are summarized as follows:

$f_m = 1206.259235$, $s = 0.797256$

$k_{1m} = -1.725762e\text{-}7$, $k_{2m} = 7.867127e\text{-}13$

The sequence of images digitized for the programmed motion are shown in the left column of Figure 5.9. Like in Section 5.1, these images are passed through a Sobel operator to extract the edge images, as shown in the right column of Figure 5.9. The plate is detected by looking for the quadrilateral of four lines. The first two rows of points in the quadrilateral are used with their corresponding model points to recover the location and motion.

According to the formulation in Chapter 4, the unknown vector b(k) has 12 parameters (6 for location and 6 for motion between two frames). On the other hand, the measurement vector d(k) has 28 elements, as two rows of 7 points are used as correspondences. The sequence of the parameters in b(k) are as follows: $\theta_x(k)$, $\theta_y(k)$, $\theta_z(k)$, $p_x(k)$, $p_y(k)$, $p_z(k)$, $\Delta\theta_x(k)$, $\Delta\theta_y(k)$, $\Delta\theta_z(k)$, $\Delta p_x(k)$, $\Delta p_y(k)$, and $\Delta p_z(k)$. Upon entry, the motion parameters in b(1|0) are set to zero, which is the most reasonable one, given no *a priori* knowledge of the motion. The location parameters in b(1|0) are set to the solution of the Pn localization problem, as discussed in Chapter 3. The Pn localization algorithm is a linear pseudo-inverse method, since the number of point correspondences are more than six. As mentioned in Chapter 3, the Pn localization problem has a unique solution if $n \geq 6$.

In tuning the EKF for this experiment, the covariance matrix, P(1|0) upon entry, is set to be a diagonal matrix with diagonal elements: 1.0e4, 1.0e4, 1.0e4, 1.0e4, 1.0e4, 1.0e4, 1.0e7, 1.0e7, 1.0e7, 1.0e9, 1.0e9, and 1.0e9. The matrix Q(k) is set to be a constant diagonal matrix with elements: 1.0e3, 1.0e3, 1.0e3, 1.0e3, 1.0e3, 1.0e3, 1.0e3, 1.0e3, 1.0e3, 1.0e5, 1.0e5, and 1.0e5. In this experiment, the covariance, R(k), is set to be an identity matrix. The values selected here is the result of several tries, and the heuristic rules for tuning are to be discussed later.

The recovered location and motion parameters are converted back to the reference coordinates, and compared with the programmed values in Table 5.1. The errors of estimation are summarized in Figure 5.10. As can be seen, the estimation of the location part, $\theta_z^r(k)$ and $p_x^r(k)$, converges at the first frame. The error for $p_x^r(k)$ is less than 1.0 mm after frame 3. The estimation of motion, $\Delta\theta_z^r(k)$ and $\Delta p_x^r(k)$, converge at frame 2, because motion can only be recovered after, at least, two images are taken. The error for $p_x^r(k)$ is less than 3.0 mm. after the third frame, and eventually goes below 1.0 mm. The estimation of $\theta_z^r(k)$ and $\Delta\theta_z^r(k)$ is not as accurate as that of $p_x^r(k)$ and $\Delta p_x^r(k)$, probably because the target (i.e. the plate) is far away from the camera (about 1.0 meter), and when it rotates, the position of the points on two succesive images do not change much. Consequently, the noise introduced in measuring the point coordinates in the image and the camera model error have more effect on $\theta_z^r(k)$ and $\Delta\theta_z^r(k)$ than on $p_x^r(k)$ and $\Delta p_x^r(k)$. In real applications, when the object is far away, its orientation is not of much interest. For example, it is not until a manipulator is going to pick up the object that the robot needs to know the orientation of the object. To have a better estimate of the orientation and incremental rotation, the robot can wait for the target to come closer, or else zoom in the object. When the camera has a close view of the object, the estimation of orientation and incremental rotation is as good as that for position and incremental translation.

When it comes to tuning the filter, experience is of great importance. Several sets have to be tried before an acceptable one comes out. The set of P(1|0) and Q(k) used here are by no mean the best one. However, there are heuristic rules for tuning the filter. The initial value of the error covariance, P(1|0), is one of the factors that affect the step size of the update at the initial stage. As time evolves, P(k|k) decreases, thus leaving the update step size to be determined by the ratio Q(k)/R(k). Obviously, a small step size will slow down the rate of convergence, while a large step size may make the estimate too sensitive to the measurement noise. Therefore, a compromise has to be made to choose a suitable set of filter parameters. In practice, a non-zero Q(k) has to be used to ensure that

P(k|k) will not decrease to zero as k increases, otherwise the filter may stop taking in the measurements to update the estimation. Based on these rules, the algorithm has to be tuned in its installation stage, or learning stage. No attempt in this thesis was made to work out a self-tuning procedure for the learning stage; however, it is a desirable goal to be attained.

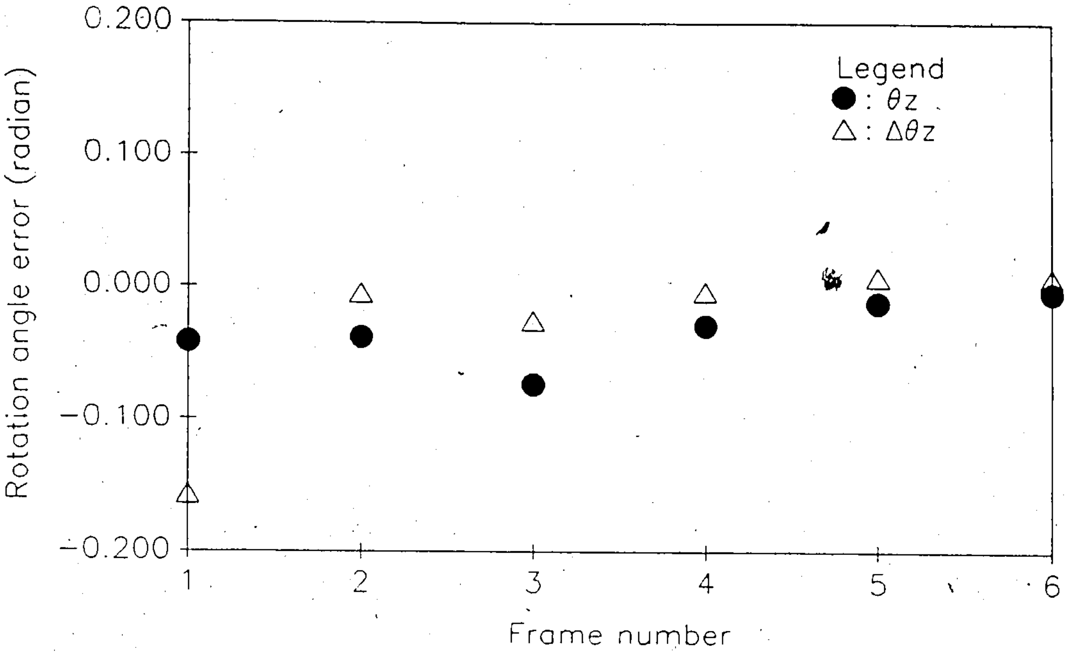
As far as the global solution is concerned, the EKF procedure is inherently a nonlinear search method, thus a local minimum solution may be found by the EKF. However, as discussed in the chapter on object localization, when more than six point or line correspondences are used, a unique solution of location can be guaranteed. When using EKF to recover motion, it is suggested that the unique solution from Ln, Pn, or En localization problem be found as initial condition so that the global solution can be found by the EKF procedure.

In this particular experiment, however, two possible solutions can be found for the plate location, because the points are symmetric with respect to the rotation axis of θ_z^p, as shown in Figure 5.1. Consider the case when the plate is parallel to the focal plane and is far away from the camera. Let us take two images of the plate: one with $\theta_z^p = 9$ degrees and the other with $\theta_z^p = -9$ degrees. These two images look almost the same. Particularly, under measurement noise, one can be mistaken for the other. Hence, the estimation in this experiment may sometime come up with the alternative, but incorrect solution.

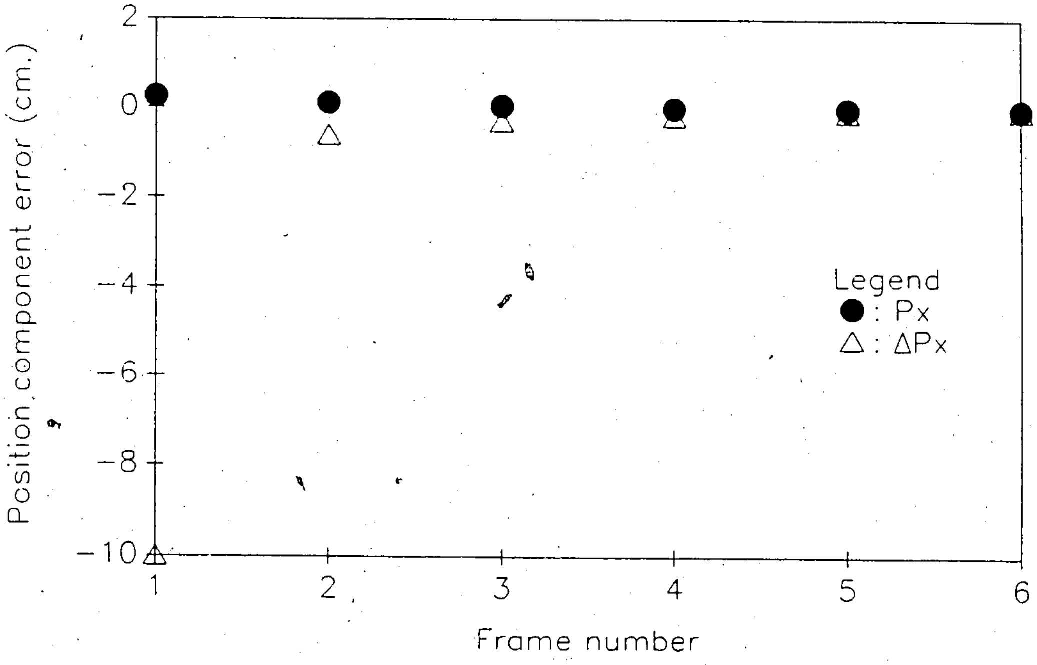
Figure 5.9 The sequence of images used in the motion recovery experiment. The left column contains the digitized intensity images of the plate at the six locations shown in table 5.1. The right column contains corresponding edge images extracted with the Sobel operator.

[redacted]

Figure 5.10 The errors in motion recovery. The errors are compared in the reference coordinates upon which the programmed motion (Table 5.1) is defined. (a) Estimation errors of rotation angles. (b) Estimation errors of position components.



(a) Estimation errors of rotation angles



(b) Estimation errors of position components

Chapter Summary. In this chapter, we have seen three implemented subsystems for a partial realization of the methods proposed in this thesis. The camera calibration subsystem calculates the focal length to be used by the other two subsystems; the calibration is done off line. The object recognition and localization subsystem is a partial realization of the proposed "locating-labeling" paradigm. Only line primitives were used in the implementation of the recognition subsystem. The motion recovery subsystem was completed a year earlier than the recognition and localization subsystem. So far, These two subsystems have yet to be integrated. However, from these partial implementations, the effectiveness of the proposed method for 3D object recognition, localization, and motion recovery has been demonstrated.

Chapter 6. Conclusions

The use of conic primitives (ellipse, hyperbola, line, and point) in object localization and motion recovery has been analytically studied, and closed-form solutions have been obtained for the object localization problem from conic correspondences. As shown in Chapter 3, the minimum number of elliptic or hyperbolic correspondences required to locate an object in 3D is one, and that of line or point correspondences is three. Due to the presence of noise in the image and in the primitive extraction process, the location determined from a mimimum number of conic correspondences may not always be very accurate. However, the ability to determine the object location from a minimum number of correspondence provides us information to establish additional correspondences with little effort. As more consistent correspondences are established, more accurate estimates of the object location can then be obtained.

The closed-form solutions for locating an object in 3D from a minimum number of conic primitives are applied to the recognition of 3D objects from a single intensity image. The ability to determine the object location "early" in the recognition process has been demonstrated in Section 5.2 by the partial implementation of only line primitives (though the same priciples applied to all classes of conic primitives) to provide a great reduction in computational complexity. This demonstration shows that the proposed "locating-labelling" paradigm is not computationally intensive since branches of unnecessary matching tests can be avoided if the object location is known early, and the computation of the object location is in closed-form. An analytical explanation of this reduction in computational complexity has been given in Section 2.3.

The locating-labelling paradigm divides the recognition (or corresponding) process into two stages. The first is to formulate a minimum number of hypothesized matches sufficient to locate the object. The second is to test and augment the set of hypothesized matches until the set of matches is rejected as inconsistent or is sufficient to uniquely determine the object's identity.

In order to reduce the search space in formulating a minimum number of correspondences at the initial stage, image feature groups (e.g., a quadrilateral of four lines) are detected and used to index the likely object models and to establish probable hypotheses. Like most other recognition schemes, this initial stage uses invariant properties and structural relations to limit the search space. A detailed discussion of this stage has been given in Section 2.3.

In the second stage, the location of the object is hypothetically determined from the minimum set of hypothesized matches. The set of matches is then tested immediately against the estimated location. If the set of hypothesized matches is tested to be consistent with the recovered location, additional hypothesized matches can be easily obtained by projecting an unmatched model primitive (which must be visible at the estimated location) and by pairing it to image primitives which have attributes close to those of the projected model primitive. In other words, the second stage of testing and expanding the set of hypothesized matches consists of four cyclic processes: localization from hypothesized matches, consistency testing of hypothesized matches with the recovered location, prediction of image primitives using the recovered location and the hypothesized object model, and augmentation of hypothesized matches. Augmenting the set of hypothesized matches is strictly limited in search space, as hypothesized matches inconsistent with the hypothetically recovered object location are quickly detected and rejected.

When the set of hypothesized matches is large enough to confirm recognition, the recognition process terminates, and the accurate 3D location of the recognized object is determined from all matches. Further, the 3D motion of a recognized object from the previous image is recovered by using an Extended Kalman Filter approach. The recovered location and motion as well as recognition are important in tasks such as picking objects from conveyors or other transportation devices.

It is worth noting that an object localizable by the proposed "locating-labelling" method need not have all its parts as conic components. As long as some parts of the object consist of conic segments, the proposed method will work by first focusing on those conic segments and later verifying the recognition with non-conic segments.

The images used here are intensity images taken by a video camera. The results show that "absolute" 3D orientations and positions of objects can be recovered by using a static image. Further, 3D motion parameters (three for rotation and three for translation) between two frames can be recovered.

Also, by working on an intensity image alone, 3D object recognition is possible and economical, if location information is properly used. However, this is not to say that intermediate representations such as $2\frac{1}{2}$D sketches and intrinsic images are not important (a $2\frac{1}{2}$D sketch usually contains explicit infomation about depths or local surface orientations, while a 2D sketch does not). The point to be taken here is on the proper use of location information rather than on the level of representation.

For situations where intermediate representation is inevitably needed (e.g., range data is usually needed in autonomous navigation), this thesis makes two contributions. First, the proposed method for recognition using intensity images can serve as an initiation process or as a verification process in object recognition, as is illustrated in Figure 6.1. Second, the extension of closed-form locating algorithms can be made for intermediate representations; for example, for quadrics. Thus, the locating-labelling paradigm can be used with both intermediate representation and intensity images, as shown Figure 6.2.

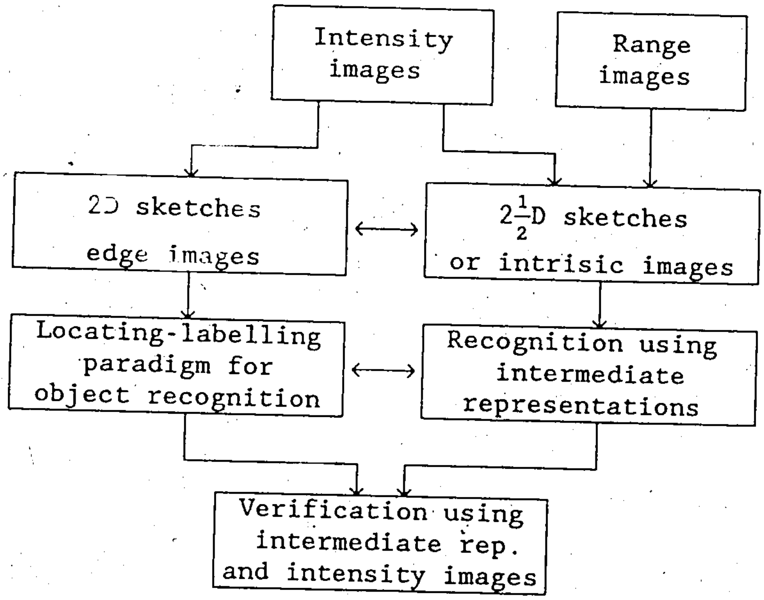

Intensity images
Range images
2D sketches edge images
$2\frac{1}{2}$D sketches or intrisic images
Locating-labelling paradigm for object recognition
Recognition using intermediate representations
Verification using intermediate rep. and intensity images

Figure 6.1 3D object recognition by using locating-labelling paradigm with 2D sketches and other recognition schemes with $2\frac{1}{2}$D sketches.

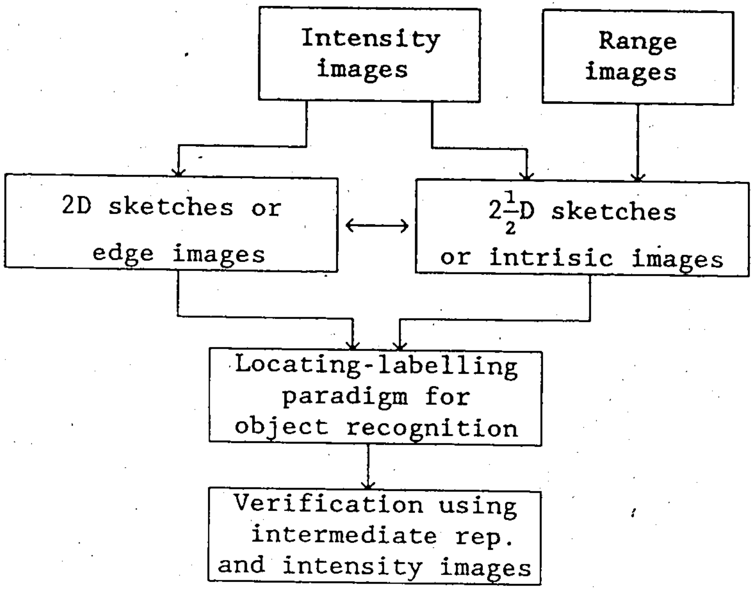

Intensity images
Range images
2D sketches or edge images
$2\frac{1}{2}$D sketches or intrisic images
Locating-labelling paradigm for object recognition
Verification using intermediate rep. and intensity images

Figure 6.2 3D object recognition by using locating-labelling paradigm with both 2D and $2\frac{1}{2}$D sketches.

In the following, we briefly compare the proposed object recognition method with other recent methods based on a single intensity image.

Comparison With ACRONYM. The proposed method differs from ACRONYM by Brooks (1981) in that ACRONYM infers constraints from sensed cues and the allowable intervals of free parameters, and then hypothesizes matches of model primitives to image primitives. The hypothesized matches are tested by propagating the associated constraints over the free parameters to see if the set of constraints is satisfiable or consistent.

Another difference is in the task domain. Although ACRONYM was modeled with 3D objects, the system was tested with aerial photographs of airplanes on the ground. The objects in aerial photographs can be regarded as 2D in nature, since little variation in depth for different object parts can be discerned. Indeed, it has been argued by Huttenlocher and Ullman (1987) that ACRONYM can only recognize 2D objects from 2D images. In contrast, the proposed method is capable of recognizing 3D objects from a single 2D image.

Comparison With Alignment Method. The proposed method differs from the method by Huttenlocher and Ullman (1987) in two ways. First, their method uses a parallel projection with a scale factor to approximate the perspective projection, whereas the proposed method uses a direct perspective projection. This difference can be significant when all parts on the object do not have approximately the same distance from the view point. In this case, foreshortening is different for different parts. The difference becomes severe when the object is close to the viewpoint. The second difference is that only critical points (zero crossings of curvature or inflection points in the contour) are used in the method by Huttenlocher and Ullman, whereas lines and conics, in addition to critical points, are formulated in this proposed method. Though high curvature points in the image can correspond to the high curvature points in the object, the measurement of such points may not always be stable (Fisher and Bolles, 1986). In particular, the use of the derivative for curvature is always subject to noise, although a certain amount of smoothing is usually used. The measurement of attributes of lines and conics are more reliable than that of critical points, because attributes of lines or ellipses are global and subject to less noise, compared to attributes of a point. For example, the slope of an image line is measured by using all points on the line, thus the slope is quite reliable despite the fact that noise can be present at each point.

Comparison With SCERPO. Like the approach by Huttenlocher and Ullman, Lowe's SCERPO (1987) uses a single type of primitive --- line segments, in contrast to the generic conic primitives proposed (though not fully implemented) in this thesis. In fact, the objects have to be polyhedral to be recognizable in Lowe's SCERPO. Further, the proposed approach uses a different locating algorithm with line correspondences. Although Lowe formulates the localization problem in a novel way, the problem remains nonlinear, and a Newton search method is needed to estimate the six unknown parameters. As has been shown in Chapter 3, three line correspondences can produce four stable solutions for location parameters. However, which one of the four global solutions will be found by the Newton method used in SCERPO will depend on the initial guess of the parameters. Worse still, a local minimum may be found instead of a global minimum. When a local minimum solution or one of the incorrect global solutions is found for a set of consistent matches, the set of matches may be tested as inconsistent and rejected. Another numerical difference in using line correspondences for localization is that SCERPO minimizes the averaged equation errors of slope and Y-intercept to obtain the location parameters. It can be readily shown that the equation error of line slope is non-uniformly distributed over slant angles of the line. With such an uneven distribution of equation errors, the Newton method may produce solutions favouring certain lines. For example, a perturbation of slant angle at higher slope values will produce larger errors than at lower slope values; thus, the Newton method may find a solution in favor of lines with large slope values.

Despite the advantage of this proposed method over SCERPO in localization algorithms, another major contribution of Lowe's SCERPO, namely Perceptual Organization for initiating hypothesized matches, is still unparalleled by this proposed method. It is believed that proper use of Lowe's Perceptual Organization for formulating probable sets of minimum hypothesized matches (at the first stage of matching process) and the use of the locating algorithms derived in this thesis for testing and augmenting each set of hypothesized matches (at the second stage of matching process) can lead to a recognition system with better performance than either IRLM or SCERPO.

Scope of Further Research. Our first concern is is to extend the scale of implementation; that is, to include the already formulated ellipse and hyperbola primitives into the recognition subsystems and to include overlapping objects in the scene. Further, we need to improvement the first stage of matching (that is, initiating probable sets of minimum hypothesized matches) by employing such techniques as Lowe's Perceptual Organization. Also, the the recognition subsystem must be intergrated with the motion recovery subsystem before this whole system can be fully appreciated.

References

Anderson, B. D. O. and Moore, J. B. (1977). Optimal Filtering. Prentice-Hall.

Balakrishnan, A. V. (1984). Kalman Filtering Theory. Optimization Software.

Ballard, D. H. and Brown, C. M. (1982). Computer Vision. New Jersey: Prentice-Hall, Inc.

Barnard, S.T. (1983). Interpreting Perspective Images. Artificial Intelligence 21. 435-462.

Besl, P. J. and Jain, R. C. (1985). Three Dimensional Object Recognition. Computing Surveys, 17(1).

Besl, P. J. and Jain, R. C. (1986). Invaraint Surface Characteristics for 3D object Recognition in Range images. Computer Vision, Graphic, and Image Processing 33. 33-80.

Bischof, W.F. and Ferraro, M. (1987). Curved Mondrians: A generalized Approach to Shape form Shading. (Technical Report No.87-06). Alberta Centre for Machine Intelligence and Robotics, University of Alberta, Edmonton, Alberta.

Bolle, R.M. and Cooper, D.B. (1986). On Optimally Combining Pieces of Information, with Application to Estimating 3D Complex-Object Position from Range-Data. IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol. PAMI-8, No.5. 619-638.

Bolles, R.C. and Fischler, M.A. (1981). Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated cartography. Communication of the ACM, Vol.24, No.6. 381-395.

Bolles, R.C. and Gain, R.A. (1982). Recognising and Locating Partially Visible Objects: The Local-Feature-Focus-Method. The International Journal of Robotics Research, 1(3).

Bolles, R. C. and Horaud, P. (1984). 3DPO: A Three Dimensional Part Orientation System. The International Journal of Robotics Research, 5(3).

Brooks, M.J. and Horn, B.K.P. (1985). Shape and source from shading. Proceedings of the 9th International Joint Conference on Artificial Intelligence, Los angeles: Morgan Kaufman. 932-936.

Brooks, R. (1981). Symbolic Reasoning Among 3-D Models and 2-D Images. Artificial Intelligence 17. 185-348.

Chin, R. T. and Dyer, C. R. (1986). Model Based Recognition in Robot Vision. Computing Surveys, 18(1).

Davis, E. (1987). Constraint Propagation with Interval Labels. Artificial Intelligence 32. 281-331.

Durrant-Whyte, H. F. (1986). Consistent Integration and Propagation of Disparate Sensor Observation. Proceedings of IEEE Conference on Robotics and Automation (pp. 1464-1469). San Francisco, California.

Faugeras, O.D., Ayache, N., and Faverjon B. (1986). Building Visual maps by Combining NoisyStereo Measurements. Proceedings of IEEE Conference on Robotics and Automation (pp.1433-1438). San Francisco, California.

Faugeras, O. D., and Hebert, M. (1986). The Representation, Recognition, and Locating of 3-D Objects. The International Journal of Robotics Research, 5(3), 27-52.

Fischler, M.A. and Bolles, R.C. (1986). Perceptual Organization and Curve Partitioning. <u>IEEE Transaction on Pattern Analysis and Machine Intelligence</u>, Vol.PAMI-8, No.1. 100-105.

Foley, J. D. and Van Dam, A. (1981). <u>Fundamentals of Interactive Computer Graphics</u>. Addison-Wesley.

Gibbson, J.J. (1979). <u>The ecological appraoch to visual perception</u>. Boston: Houghton Mifflin.

Grimson, W.E.L. and Lozano-Pérez, T. (1984). Model-based Recognition and Localization from sparse range or Tactile data. In A. Rosenfeld (ED.), <u>Techniques for 3-D Machine Perception</u> (pp.113-148). Elsevier Science Publishers B.V. (North-Holland), 1986.

Grimson, W.E.L. and Lozano-Pérez, T. (1985). <u>Recognition and Localization of overlapping parts from sparse range</u>. (A.I. Memo No.841). Artificial Intelligence Laboratory, MIT.

Guzman, A. (1968). Decomposition of A Visual Scene into Three-Dimensional Bodies. <u>AFIDS Fall Joint Conference 33</u>. 291-304.

Hildreth H. C. and Hollerback, J. M. (1985). <u>The Computational Approach to Vision and Motor Control</u>. (A.I. Memo No. 846). Artificial Intelligence Laboratory, MIT.

Horaud, P. and Bolles, R. C. (1984). 3DPO: A System for Matching 3D Objects in Range Data. <u>IEEE 1984 Conference on Robotics</u>.

Horowitz, E. and Sahna, S. (1977). <u>Fundamentals of Computer Algorithms</u>. Computer Sciense Press, Inc.

Horn, B.K.P. (1977). Understanding image intensities. <u>Artificial Intellisgence 8</u>. 201-231.

Hummel, R.A and Zucker, S.W. (1983). On the foundations of relaxation laeling process. IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol. PAMI-5. 267-287.

Huttenlocher, D.P. and Ullman, S. (1987). Recognizing Rigid Objects by Aligning Them with an Image. (A.I. Memo No. 937). Aritificial Intelligence Laboratory, MIT.

Iberall T., Bingham, G. and Arbib, M. A. (1985). Opposition Space as a Structuring Concept for the Analysis of Skilled Hand Movements. COINS Technical Report 85-19. Laboratory for Perceptual Robotics, Department of Computer and Information Science,University of Massachusetts, Amherst, Massachusetts.

Korn, G. and Korn, T. (1961). Mathmatical handbook for Scientists and Engineers. McGraw-Hill Company, Inc.

Kanade, T. (1981). Recovery of the Three-Dimensional Shape of an Object form a Sigle View. Artificial Intelligence 17. 409-460.

Lee, H.C. and Fu, K.S. (1983.a). Generating Object Description for Model Retrieval. IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol. PAMI-5, No.5. 462-471.

Lee, H.C. and Fu, K.S. (1983.b). 3D Shape from Contour and Selective Confirmation. Computer Vision, Graphic, and Image Processing 22. 117-193.

Levin, J. (1976). A Parametric Algorithm for Drawing Pictures of Solid Objects Composed of Quadric Surfaces. Communication of the ACM, Vol.19, No.10. 555-563.

Levin, J. (1979). Mathematical Models for Determining the Intersections of Quadric Surfaces. Computer Vision, Graphic, and Image Processing 11. 73-87.

Levine, M.W. and Shefner, J.M. (1981). Fundamentals of Sensation and Perception. Addison-Wesley Publishing Company.

Liao, Y. (1981). A two-stage method of fitting conic arcs and straight-line segments to digitized contours. Proceedings Pattern Recognition and Image Processing Conference, Dallas, Texas. 224-229.

Lowe, D. G. (1987). Three-Dimensional Object Recognition from Single Two-Dimensional Images. Artificial Intelligence 31. 355-395.

Mackworth, A. K. and Freuder E. C. (1985). The complexity of some polynomial network consistency algorithms for constraint satisfication problems. Artificial Intelligence 25. 65-74.

Marr, D. and Poggio, T. (1976). Cooperative Computation of Stereo Disparity. Science 194. 283-287.

Marr, D. (1982). Vision. San Francisco: Freeman.

Maybeck, P. S. (1982). Stochastic Models, Estimation, and Control.Academic Press (Vols. 1-2).

Macburney, D.H. and Collings, V.B. (1983). Introduction to Sensation / Perception. Pretice Hall.

Meirovitch, L. (1970). Methods of Analytical Dynamics.

Nillsson, N. J. (1980). Principles of Artificial Intelligence.

Mulgaonkar, P.G. and Shapiro, L.G. (1985). Hypothesis-Based Geometric Reasoning about Perspective Images.

Oshima, M and Shirai, Y. (1983). Object Recognition Using Three-Dimensional Information. IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol. PAMI-5, No.4. 353-361.

Paul, R. (1981). Robot Manipulators: Mathematics, Programming, and Control MIT Press.

Pearl, J. (1984). Heuristics: Intelligent Search Strategies for Computer Problem Solving. Anderson-Wesley Publishing Co.

Rosen, C.A. (1979). Machine Vision and Robotics: Industrial Requirements. In C.G. Dodd and L.Rossol (ED.), Computer Vision and Sensor-Based Robots (pp.3-22). General Motors Research Laboratories.

Pentland, A.P. (1986). Perceptual Organization and the Representation of Natural Form. Artificial Intelligence 28, 293-331.

Pentland, A.P. (1987). Recognition by Parts. (Technical Notes No. 406). SRI international.

Pentland, A.P.and Bolles, R. (1987). Learning and Recognition in Natural Environments. (Technical Notes). SRI international.

Rektorys, K. (1969). Survey of applicable Mathematics. The MIT Press.

Rich, E. (1983). Artificial Intelligence. McGraw-Hill Book Company.

Robert, L.G. (1965). Machin Perception of Three-Dimensional Solids. In J.P. Tippett *et al.* (Ed.). Optical and Electro-Optical information processing(pp.159-197). Cambridge, MA: MIT Press.

Rosenberg, R.M. (1977). Analytical Dynamics of Discrete Systems. New York: Plenum Press.

Sarraga, R.F. (1983). Algebraic Methods for Intersections of Quadratic Surfaces in GMSOLD. Computer Vision, Graphic, and Image Processing 22: 222-238.

Stuelpnagel, J. (1964). On the parametrization of the Three-Dimensional Rotation group. SIAM Review, Vol.6, No.4, 422-430.

Tenenbaum, J.M., Barrow, H.G., and Bolles, R.C. (1979). Prospects for Industrial Vision. In C.G. Dodd and L.Rossol (ED.), Computer Vision and Sensor-Based Robots (pp.239-259). General Motors

Research Laboratories.

Tsai, R. Y. (1986). An efficient and Accurate Camera Calibration Technique for 3D Machine Vision. Proceedings of IEEE Conference on Computer Vission and Pattern Recognition(pp.364-374).

Ullman, S. (1986). An Approach to Object Recognition: Aligning Pictorial Description. (A.I. Memo No. 931). Artificial Intelligence Laboratory, MIT.

Waltz, D. (1975). Generating semantic descriptions from drawing s of scences with shadows. In P. Winston (Ed.). The psychology of computer vision. New York: McGraw-Hill.

Widrow, B., and Stearns, S. D. (1985). Adaptive Signal Processing. Prentice-Hall.

Wittenburg, J. (1977). Dynamics of Systems of Rigid Bodys. B.G. Teubner, Stuttgart.

Woodham, R.J. (1978). Photometric Stereo: A Reflectance Map Technique for Determing Surface Orientation from Image Intensity. (A.I. Memo No. 479). Artificial Intelligence Laboratory, MIT.

Wu, J., Rink, R.E., Caelli, T.M., and Gourishankar, V. (1988). Recovery of 3D locations and motions of rigid objects through Camera Imaging (Extended Kalman filter approach). International Journal of Computer Vision, (in press).

Wu, J. and Caelli, T.M. (1988). IRLM: An Imaging System for Object Recognition, Localization, and Motion Recovery. Proceeding of Vision Interface 1988. Edmonton, Canada.

Appendix A. Perspective Projection of Conic Sections

This appendix gives a mathematical proof for Table 2.3 in Chapter 2, assuming that the viewed conic section is in front of the camera and the focal length is f. The cases of point and line are trivial and omitted here. The cases of ellipse, hyperbola, parabola, are treated in different sections. The formula used in the proof will be used again in Appendix D.1 to determine the location of the object once correspondences between image and model conics are established.

A.1 Perspective Projection of an Ellipse

Let the ellipse be the intersection of $x^2/\alpha^2 + y^2/\beta^2 = 1$ and $z = 0$, where x, y, and z are in the primitive coordinate basis, and let the transformation of primitive coordinate basis to the camera coordinate basis be described by G and t, one gets:

$$\begin{bmatrix} x^c \\ y^c \\ z^c \end{bmatrix} = G \begin{bmatrix} x \\ y \\ 0 \end{bmatrix} + t \qquad \text{(A.1.1)}$$

where (x,y) represent a point in the primitive coordinate basis, and (x^c, y^c, z^c) is the same point in camera coordinate basis.

Multiplying each side with G^T and using $X = -fx^c/z^c$ and $Y = -fy^c/z^c$ from (1.1) yields:

$$\begin{bmatrix} x \\ y \\ 0 \end{bmatrix} + G^T t = G^T \begin{bmatrix} x^c \\ y^c \\ z^c \end{bmatrix} = -z^c/\ f\ G^T \begin{bmatrix} X \\ Y \\ -f \end{bmatrix} = -z^c/\ f\ G^T\ D^T\ w \qquad \text{(A.1.2)}$$

where $D = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -f \end{bmatrix}$, $\mathbf{w} = \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$, and (X,Y) represents the corresponding image point.

Let $u = G^T t$ and g_i be the i-th column in D matrix. From the third element of the vector in the previous equation, one has

$$-z^c = f\, u_3 \,/\, (g_3^T D^T \mathbf{w}), \quad \text{if } (g_3^T D^T \mathbf{w}) \neq 0 \qquad \text{(A.1.3)}$$

Substituting (A.1.3) into (A.1.2) yields

$$x = u_3\, (g_1^T D^T \mathbf{w}) \,/\, (g_3^T D^T \mathbf{w}) - u_1 \qquad \text{(A.1.4)}$$

$$y = u_3\, (g_2^T D^T \mathbf{w}) \,/\, (g_3^T D^T \mathbf{w}) - u_2 \qquad \text{(A.1.5)}$$

Using the ellipse equation $x^2/\alpha^2 + y^2/\beta^2 = 1$, one obtains

$$\mathbf{w}^T D\ G\ B \begin{bmatrix} \alpha^{-2} & 0 & 0 \\ 0 & \beta^{-2} & 0 \\ 0 & 0 & -1 \end{bmatrix} B^T G^T D^T \mathbf{w} \equiv \mathbf{w}^T A\ \mathbf{w} = 0 \qquad \text{(A.1.6)}$$

where $B = \begin{bmatrix} u_3 & 0 & 0 \\ 0 & u_3 & 0 \\ -u_1 & -u_2 & 1 \end{bmatrix}$

It can be easily shown that $\det(A) = -f^2 u_3^4/(\alpha^2\beta^2) \leq 0$. Thus, if $u_3 = (-z^c/f)\ (g^T D^T \mathbf{w}) \neq 0$, then $\det(A) \neq 0$, and the projection of the ellipse on the image plane is a proper or regular conic section (Korn, 1961). Whether the image of an ellipse is an ellipse, a hyperbola, or a parabola will depend on the sign of $a_{11}a_{22} - a_{12}^2$, where a_{ij} is the (i,j) element in the A matrix. It can be shown that

$$\begin{aligned} a_{11}a_{22} - a_{12}^2 &= (f\ u_3)^2 (t_3^2 - (\alpha\ g_{31})^2 - (\beta\ g_{32})^2) \,/\, (\alpha\beta)^2 \\ &\geq (1\ u_3)^2 (t_3^2 - (\max(\alpha^2, \beta^2)) \,/\, (\alpha\beta)^2 \end{aligned} \qquad \text{(A.1.7)}$$

In order for the ellipse to be visible to the camera, $-t_3$ has to be positive. Consequently, $-t_3 > \max(\alpha,\beta)$ is the sufficient condition for the projection of the ellipse to be an image ellipse. The physical meaning of this condition is that the ellipse is completely in front of the camera.

When $u_3 = (-z^c/f)(g_3^T D^T \mathbf{w}) = 0$, one has $g_3^T D^T \mathbf{w} = 0$ since $-z^c$ is positive for every point on the ellipse. As a result, the image of the ellipse is an image line described by

$$g_{13} X + g_{23} Y - f\, g_{33} = 0 \tag{A.1.8}$$

A.2 Perspective Projection of a Hyperbola

Let the hyperbola be the intersection of $x^2/\alpha^2 - y^2/\beta^2 = 1$ and $z = 0$, where x, y, and z are in the primitive coordinate basis. Following Appendix A.1, one derives the following equation for the projected hyperbola:

$$\mathbf{w}^T D\ G\ B \begin{bmatrix} \alpha^{-2} & 0 & 0 \\ 0 & -\beta^{-2} & 0 \\ 0 & 0 & -1 \end{bmatrix} B^T G^T D^T \mathbf{w} = \mathbf{w}^T A\ \mathbf{w} = 0 \tag{A.2.1}$$

Similarly, $\det(A) = f^2\, u^4 / (\alpha^2\beta^2) \geq 0$, and

$$a_{11}a_{22} - a_{12}^2 = -(f\, u_3)^2 (t_3^2 - (\alpha\, g_{31})^2 + (\beta\, g_{32})^2) / (\alpha\beta)^2$$
$$\leq (f\, u_3)^2 (t_3^2 - \alpha^2) / (\alpha\beta)^2 \tag{A.2.2}$$

The sufficient condition for $a_{11}a_{22} - a_{12}^2$ to be negative (so that the image of a hyperbola is an hyperbola) is that $-t_3 > \alpha$ and $u_3 \neq 0$. In practice, a hyperbola has limited range, and if it is completely in front of the camera, then the image of such hyperbola is also a hyperbola. Again, when $u_3 = 0$, the image of such hyperbola is an image line described by (A.1.8).

A.3 Perspective Projection of a Parabola

Let the parabola be the intersection of $y = \alpha x^2$ and $z = 0$, where x, y, and z are in the primitive coordinate basis. Following Appendix A.1, one obtains:

$$\mathbf{w}^T D\ G\ B \begin{bmatrix} -\alpha & 0 & 0 \\ 0 & 0 & 1/2 \\ 0 & 1/2 & 0 \end{bmatrix} B^T G^T D^T \mathbf{w} = \mathbf{w}^T A\ \mathbf{w} = 0 \qquad \text{(A.3.1)}$$

Similarly, $\det(A) = \alpha\ f^2 u_3^4/4$, and

$$a_{11}a_{22} - a_{12}^2 = (f\ u_3)^2(-\alpha\ t_3 g_{32} - g_{31}^2/4\ ) \qquad \text{(A.3.2)}$$

Since the sign of $a_{11}a_{22} - a_{12}^2$ is indeterminate, the image of a parabola can be an ellipse, a hyperbola, or a parabola depending on the orientation and position of the parabola relative to the camera. In practice, the parabola has limited range; therefore, its image ellipse or hyperbola will have a missing boundary segment. Again, when $u_3 = 0$, the image of a parabola is an image line described by (A.1.8).

Appendix B. Finding Constraints for Constraint Propagation

By parameterizing the orientation matrix with quaternions, equations describing the relations such as limits of viewing window and correspondences of image and model primitives usually become quadratic in the quaternion q and the position vector **p**. Here we derive some of these quadratic constraints and discuss their use in matching image and model primitives. To use the constraints, a way to find the intervals of the variables from the quadratic constraints has to be found. This is usually called constraint propagation, and a way to propagate quadratic constraints over their variables is derived in next appendix.

Detailed discussion of parameterizing rotation, including using quaternions, is given in Chapter 3. For convenience, a summary of the quaternion expression is duplicated here. The quaternion expression uses the rotation axis (around which the rotation takes place) and the rotation angle (the amount of rotation) to parameterize the orientation matrix as follows:

$$O = \begin{bmatrix} q_1^2 - q_2^2 - q_3^2 + q_4^2 & 2(q_1q_2 - q_3q_4) & 2(q_1q_3 + q_2q_4) \\ 2(q_1q_2 + q_3q_4) & -q_1^2 + q_2^2 - q_3^2 + q_4^2 & 2(q_2q_3 - q_1q_4) \\ 2(q_1q_3 - q_2q_4) & 2(q_2q_3 + q_1q_4) & -q_1^2 - q_2^2 + q_3^2 + q_4^2 \end{bmatrix} \qquad (B.1)$$

with the constraint that $q_1^2+q_2^2+q_3^2+q_4^2 = 1$. With the four parameters, the vector, $[q_1, q_2, q_3]^T$, defines the rotation axis, and $2\sin^{-1}(q_4)$ indicates the rotation angle.

Before constraints can be derived, the relation between a point in space and its counterpart in the image need to be mentioned. The relation between coordinates in camera space and coordinates in object space is given by:

$$r^c = [x^c, y^c, z^c]^T = Or^o + p \tag{B.2}$$

where O and p are the orientation matrix and position vector of the object coordinate basis relative to the camera coordinate basis. r^o and r^c are the vectors in the two coordinate bases.

Furthermore, the relation of a point, r^c, in the camera coordinate basis and its image, (X,Y), on the image plane is given by the perspective projection:

$$X = -x^c/z^c, \text{ and } Y = -y^c/z^c \tag{B.3}$$

B.1 Possible Quadratic Constraints

Constraints from the Quaternion Expression

Upon entry, $q_1 \in [-1, 1]$, $q_2 \in [-1, 1]$, $q_3 \in [-1, 1]$, and $q_4 \in [-1, 1]$. In addition, the unit norm constraint gives the following inequality:

$$\mathrm{Min}(1 - \sum_{j \neq i} q_j^2) \le q_i^2 \le \mathrm{Max}(1 - \sum_{j \neq i} q_j^2)$$

Constraints from Visible Parts of Objects

If certain parts of the object is visible, then the following constraints are valid for points on the surfaces of these parts.

Depth constraints. If a point appears on the image, then it must be in front of the camera. So, the z-component of the point in camera coordinate basis must be negative. That is,

$$z^c(r^o) < 0. \tag{B.4}$$

for every visible point r^o in the object. From (B.1) and (B.2), it follows that (B.4) is quadratic in terms of q and p.

Viewing direction constraints. If the visible side of a surface is given by its normal vector n^o at the point r^o, then the surface is visible to the camera only if

$$(O\, r^o + p) \cdot n^o(r^o) < 0.$$

The inequality can be expressed as quadratic form of q and p.

Viewing Window constraints. If the visible point in the image is within the rectangular defined by (W_X^l, W_Y^l) and (W_X^h, W_Y^h), then it follows that

$$W_X^l \le -x^c(r^o)/z^c(r^o) \le W_X^h \text{ , and } W_Y^l \le -y^c(r^o)/z^c(r^o) \le W_Y^h \text{ .}$$

Since $z^c(r^o)$ is not zero, these constraints can be expressed as a conjunction of quadratic fomula for q and p by multiplying $z^c(r^o)$ to each side of the inequalities.

Constraints from Primitive Correspondences

Once a correspondence between image and model primitives is constituted, a set of algebraic equations decribing the match is obtained to limit the feasible set of solutions for qand p. Due to noise in the image, each obtained equation is supplimentted by an uncertainty term, thus resulting in inequalities.

Elliptic or Hyperbolic Constraint. This is the most powerful constraint being propagated. As will be shown in Appendix D, after a match between image and model ellipses (or hyperbolas), only two possible sets of solutions for q and p exist. That is, a match of this kind is sufficient to place the object in only two locations. So, another correspondence after this can be used to verify the interpretation right away.

Line-Intercept Constraints. The necessary condition for a model line to be matched to a given image line is

$$T = m\ (-x^c(r^o)/z^c(r^o)) + y^c(r^o)/z^c(r^o)$$

where T and m are measured Y-intercept and slope of the image line, and r^o is any point in the matched model line. Taking into account the measurement noise ξ for Y-intercept, one rewrites the constraint as:

$$T - \xi \leq m\,(-x^c(\mathbf{r}^o)/z^c(\mathbf{r}^o)) + y^c(\mathbf{r}^o)/z^c(\mathbf{r}^o) \leq T + \xi,$$

which can be expressed as quadratic form of q and p by multiplying each side of the inequality with $z^c(\mathbf{r}^o)$.

<u>Point Constraints</u>. After a match of image and model points (either critical points or intersections), one has the constraints:

$$X-\xi \leq -x^c(\mathbf{r}^o)/z^c(\mathbf{r}^o) \leq X+\xi, \text{ and } Y-\xi \leq -y^c(\mathbf{r}^o)/z^c(\mathbf{r}^o) \leq Y+\xi.$$

where (X,Y) is the image point on the image plane, and $\mathbf{r}^o$ is the model point in object coordinate basis. By multiplying each side of the inequalities with $z^c(r^o)$ one can express point constraints in quadratic form of q and p.

B.2 Constraints Propogation in Bottom-Up Matching

In bottom-up pairing, usually a significant image primitive is selected first. Then, each feasible model primitive is tried against the selected image primitive to see if the two primitives can form a correspondence consistent with other correspondences. A candidate model primitive is said to be feasible if, after the set of constraints imposed by the tentative match between the candidate primitive and the selected image primitive is propagated over the intervals of q and p, the tentatively revised intervals of q and p are not all null. The checking of nullness is easily done by checking the range of w_i in the algorithm *LoverVar* after using *Q2QC* and *QC2L* in Appendix C.

B.3 Constraint Propogation in Top-down Matching

In top-down matching, often a significant model primitive is selected and projected onto the image plane. Then each feasible image primitive is tested against the projected primitive to see if they can form a correspondence. If q and p have been determined discretely,feasible image primitives are those with measured attributes close to the projected attributes of the selected model primitive. If, however, q and p can only be determined in terms of intervals, the projected attributes of the selected model primitive can only be known within some lower bounds and upper bounds. To determine the lower and upper bounds for a certain attribute, the expression for that attribute is evaluated by using the known intervals of q and p. The algorithm *UpdateEqu* in Appendix C does just that. In such a case as projected attributes only known to be within intervals, an image primitive is said to be feasible if its measured attributes are within the intervals of their respective projected attributes.

Appendix C. Propogating Quadratic Constraints

The fact that equality/inequality equations constrain the ranges of their variables is often used to reason the possible values of particular variables under certain circumstances. This kind of inference is usually called constraint propagation with interval labels (Davis, 1987). In this appendix, closed-form solutions are derived for propagating quadratic constraints over their variables. Conversely, this appendix also discusses how to find the interval of an quadratic expression by using the known intervals of its variable

C.1 Evaluating Intervals of Variables Using Given Quadratic Constraints

To find the intervals of variables under a quadratic constraint, one first converts the quadratic constraint into its canonical form by using the algorithm *Q2QC* and then converts the canonical form into a linear constraint by using the algorithm *QC2L*. Secondly, one propagates the obtained linear constraint over the transformed variables by using the algorithm *LoverVar*. Finally, one applies the algorithm *UpdateVar* to update the intervals of the real variables by using the updated values of the transformed variables and the transformation used in the first step.

Algorithm *Q2QC*

Purpose: To convert the Quadratic Constraint to Canonical Form.

1. <u>Input Variables</u>:

$$v_i \in [v_i^l, v_i^h], \quad \text{for } 1 \le i \le n.$$

2. Input Constraint:

$$g(v_1, v_2) = v_1^T A v_1 + b^T v_1 + c^T v_2 \in [g^l, g^h]$$

where $v_1 = [v_1, v_2, \ldots, v_m]^T$, $v_2 = [v_{m+1}, v_{m+2}, \ldots, v_n]^T$, A is a mxm matrix, and b and c are vectors with m and n-m elements, respectively.

3. Transformation:

- Let the eigenvalues of A be λ_1, $\lambda_1, \ldots, \lambda_m$, and the corresponding normalized eigenvectors be h_1, h_2, $\ldots$, h_m.

Let $H = [h_1, h_2, \ldots, h_m]$ and $[\lambda_{m+1}, \lambda_{m+2}, \ldots, \lambda_n] = c^T$.

Let $u_1 = H^{-1} v_1$ and $u_2 = v_2$.

4. Output Variables:

$$u_i \in [u_i^l, u_i^h], \quad \text{for } 1 \le i \le n.$$

where $u_i^l = \sum_{j=1}^{m} \mathrm{Min}(h_{ji} v_j)$ and $u_i^h = \sum_{j=1}^{m} \mathrm{Max}(h_{ji} v_j)$, for $1 \le i \le m$, and $u_i^l = v_j^l$ and $u_i^h = v_j^h$, for $m+1 \le i \le n$.

5. Output Constraint:

$$g(u_1, u_2) = \sum_{i=1}^{m} \lambda_i (u_i - u_{i0})^2 + \sum_{i=m+1}^{n} \lambda_i u_i + d \in [g^l, g^h]$$

where $u_{i0} = (b^T h_i)$ and $d = -\sum_{i=1}^{m} \lambda_i u_{i0}^2$

Note that h_{ji} is the (j,i) element in the matrix H. The value of $\mathrm{Max}(h_{ji} v_j)$ is $(h_{ji} v_j^h)$ if $h_{ji} > 0$ or $(h_{ji} v_j^l)$ if $h_{ji} < 0$. The value of $\mathrm{Min}(h_{ji} v_j)$ is $(h_{ji} v_j^l)$ if $h_{ji} > 0$ or $(h_{ji} v_j^h)$ if $h_{ji} < 0$.

Algorithm *QC2L*

Purpose: To convert the quadratic constraint (in canonical form) to a linear constraint.

1. Input Variables:

$$u_i \in [u_i^l, u_i^h], \quad \text{for } 1 \le i \le n.$$

2. Input Constraint:

$$g(u_1,u_2) = \sum_{i=1}^{m} \lambda_i (u_i - u_{i0})^2 + \sum_{i=m+1}^{n} \lambda_i u_i + d \quad \epsilon \ [g^l, g^h]$$

3. Transformation:

Let $w_i = (u_i - u_{i0})^2$, for $1 \le i \le m$, and $w_i = u_i$, for $m+1 \le i \le n$.

4. Output Variables:

$$w_i \ \epsilon \ [w_i^l, w_i^h], \quad \text{for } 1 \le i \le n.$$

where, for $1 \le i \le m$, $w_i^l = 0$ if $u_{i0} \epsilon [u_i^l, u_i^h]$, or $w_i^l = \mathrm{Min}\{(u_i^l - u_{i0})^2, (u_i^h - u_{i0})^2\}$, else. For $m+1 \le i \le n$, $w_i^l = u_j^l$. Similarly, for $1 \le i \le m$, $w_i^h = \mathrm{Max}((u_i^l - u_{i0})^2, (u_i^h - u_{i0})^2)$, and for $m+1 \le i \le n$, $w_i^h = u_i^h$.

5. Output Constraint:

$$g(w) = \sum_{i=1}^{n} \lambda_i w_i + d \quad \epsilon \ [g^l, g^h]$$

Algorithm *LoverVar*

Purpose: To propagate an linear constrain over its variables.

1. Input Variables:

$$w_i \ \epsilon \ [w_i^l, w_i^h], \quad \text{for } 1 \le i \le n.$$

2. Input Constraint:

$$g(w) = \sum_{i=1}^{n} \lambda_i w_i + d \quad \epsilon \ [g^l, g^h]$$

3. Output Variables:

$$w_i \ \epsilon \ [\hat{w}_i^l, \hat{w}_i^h], \quad \text{for } 1 \le i \le n.$$

where $\hat{w}_i^l = \mathrm{Max}\{w_i^l, (1/\lambda_i)(g^l - \sum_{j \ne i} \mathrm{Max}(\lambda_j w_j))\}$, if $\lambda_i > 0$, or $\hat{w}_i^l = \mathrm{Max}\{w_i^l, (1/\lambda_i)(g^h - \sum_{j \ne i} \mathrm{Min}(\lambda_j w_j))\}$, if $\lambda_i < 0$. Similarly, $\hat{w}_i^h = \mathrm{Min}\{w_i^h, (1/\lambda_i)(g^h - \sum_{j \ne i} \mathrm{Min}(\lambda_j w_j))\}$, if $\lambda_i > 0$, or $\hat{w}_i^h = \mathrm{Min}(w_i^h, (1/\lambda_i)(g^l - \sum_{j \ne i} \mathrm{Max}(\lambda_j w_j)))$, if $\lambda_i < 0$.

Algorithm *UpdateVar*

Purpose: To update the variables of a qudratic constraint after *Q2QC*, *OC2L*, and *LoverVar* ~~are~~ used.

1. Input Variables:

$w_i \in [\hat{w}^l_i, \hat{w}^h_i]$, $u_i \in [u^l_i, u^h_i]$, and $v_i \in [v^l_i, v^h_i]$, for $1 \le i \le n$.

2. Output Variables:

$u_i \in [\hat{u}^l_i, \hat{u}^h_i]$ and $v_i \in [\hat{v}^l_i, \hat{v}^h_i]$, for $1 \le i \le n$.

where $\hat{u}^l_i = \text{Max}(u^l_i, u_{i0} - (\hat{w}^h_i)^{1/2})$ and $\hat{u}^h_i = \text{Max}(u^h_i, u_{i0} + (\hat{w}^h_i)^{1/2})$ for $1 \le i \le m$, and $\hat{u}^l_i = \hat{w}^l_i$ and $\hat{u}^h_i = \hat{w}^h_i$ for $m+1 \le i \le n$. Further,

$\hat{v}^l_i = \text{Max}(v^l_i, \sum_{j=1}^{m} \text{Min}(h_{ij}\hat{u}_j))$ and $\hat{v}^h_i = \text{Min}(v^h_i, \sum_{j=1}^{m} \text{Max}(h_{ij}\hat{u}_j))$

for $1 \le i \le m$, and $\hat{v}^l_i = \hat{u}^l_i$ and $\hat{v}^h_i = \hat{u}^h_i$ for $m+1 \le i \le n$.

C.2 Evaluating Quadratic Expressions for Given Variable Intervals

In contrast to the previous section, the problem considered here is to find the possible range of a qudratic expression using its known variable intervals. This is generally used in a top-down reasoning where the variable ranges are hypothesized or determined, and are used to predict a certain relation by evaluating the expression which describes the relation. The algorithm UpdateEqu does just that and is self-contained.

Algorithm *UpdateEqu*

Purpose: To evaluate the possible range of an expression by using the known intervals of its variables.

1. Input Variables:

$v_i \in [v^l_i, v^h_i]$, for $1 \le i \le n$.

2. Input Expression:

$$g(\mathbf{v}_1, \mathbf{v}_2) = \mathbf{v}_1^T A \mathbf{v}_1 + \mathbf{b}^T \mathbf{v}_1 + \mathbf{c}^T \mathbf{v}_2 \in [g^l, g^h]$$

where $\mathbf{v}_1 = [v_1, v_2, \ldots, v_m]^T$, $\mathbf{v}_2 = [v_{m+1}, v_{m+2}, \ldots, v_n]^T$, A is a mxm matrix, and **b** and c are vectors with m and n-m elements, respectively.

3. Transformation:

   Use Q2QC and QC2L to obtain the intervals :

   $$w_i \in [w_i^l, w_i^h], \text{ for } 1 \leq i \leq n,$$

   and the expression

   $$g(w) = \sum_{i=1}^{n} \lambda_i w_i + d$$

4. Output Expression:

   $$g \in [g^l, g^h]$$

   where $g^l = \sum_{i=1}^{n} \mathrm{Min}(\lambda_i w_i) + d$ and $g^h = \sum_{i=1}^{n} \mathrm{Max}(\lambda_i w_i) + d$

Appendix D. Minimum-Correspondence Localization

In this Appendix, a minimum number of correspondences are used to recover the 3D location of the viewed object. Three basic algorithms are derived; they are one-ellipse locator (*E1Loc*), three-line locator (*L3Loc*), and three-point locator (*P3Loc*). Other cases like two-line-one-point and one-line-two-point can be solved by modifying the basic algorithms, as is explained in Chapter 3.

In each of the locators, eight solutions for the orientation matrix O and position vector p can exist. However, half of them transform the object behind the camera, which is physically impossible, and so, they can be eliminated. The three locators share the same assumption that the focal length, f, of the camera is known. Each of the algorithms will be addressed separately in each section that follows. To avoid conflict of symbols, symbols defined in each section are only valid for that section, unless specified otherwise.

D.1 Localization Using One Ellipse correspondence

Let the transformation of the primitive coordinate basis to the model coordinate basis be described by the rotation matrix Q and the translation vector s, and the transformation of the model coordinate basis to the camera coordinate basis be described by the orientation matrix O and the postition vector p, then the transformation between the primitive coordinate basis and the camera coordinate basis is given by G(=OQ) and t(=Os+p). Let the model ellipse be the intersection of $x^2/\alpha^2 + y^2/\beta^2 = 1$ and $z = 0$, where x, y, and z are in primitive coordinate basis, and let $\mathbf{w} = [X,Y,1]^T$ and g_i be the i-th column in G matrix. It was shown in Appendix A that the projected ellipse is of the form:

$$\mathbf{w}^T D\ G\ B \begin{bmatrix} \alpha^{-2} & 0 & 0 \\ 0 & \beta^{-2} & 0 \\ 0 & 0 & -1 \end{bmatrix} B^T G^T D^T \mathbf{w} = \mathbf{w}^T A\ \mathbf{w} = 0 \qquad \text{(D.1.1)}$$

where $D = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -f \end{bmatrix}$ and $B = \begin{bmatrix} g_3 t^T & 0 & 0 \\ 0 & g_3^T t & 0 \\ -g_1^T t & -g_2^T t & 1 \end{bmatrix}$.

By using the orthonormal property of the rotation matrix G, it can be shown that

$$G\ B = [g_2 \times t,\ -g_1 \times t,\ g_3] \qquad \text{(D.1.2)}$$

where x is the cross product operator.

Suppose the measured image ellipse is given by $\mathbf{w}^T M\ \mathbf{w} = 0$. If the measured ellipse is the image of the model ellipse, then $A = \eta\ M$, where η is a scalar. Let $N = D^{-1} M\ D^{-1}$, one calculates its eigenvalues as λ_1, λ_2, and $-\lambda_3$, and the corresponding unit eigenvectors as e_1, e_2, and e_3; λ_1, λ_2, and λ_3 are all positive for an ellipse. Again, let $\mu^2 = \lambda_1/\lambda_3$, $\nu^2 = \lambda_2/\lambda_3$, and $E = [e_1, e_2, e_3]$, one has

$$E \begin{bmatrix} \alpha\mu & 0 & 0 \\ 0 & \beta\nu & 0 \\ 0 & 0 & 1 \end{bmatrix} = [\alpha\mu e_1, \beta\nu e_2, e_3] = \eta\ G\ B = \eta\ [g_2 \times t, -g_1 \times t, g_3] \qquad \text{(D.1.3)}$$

From (D.1.3) and the orthonormality of G, one obtains the following equations:

$$g_3 = e_3 \qquad \text{(D.1.4)}$$

$$t = \rho\ (e_1 \times e_2), \text{ since } t \cdot e_1 = 0 \text{ and } t \cdot e_2 = 0, \qquad \text{(D.1.5)}$$

$$e_2 \cdot g_1 = 0 \tag{D.1.6}$$

$$e_1 \cdot g_2 = e_1 \cdot (g_3 x\, g_1) = e_1 \cdot (e_3 x\, g_1) \tag{D.1.7}$$

$$g_1 = \frac{e_2 x\ e_1 x\ e_3}{\|e_2 x\ e_1 x\ e_3\|} = \frac{(e_2 . e_3)e_1 - (e_2 . e_1)e_3}{\|(e_2 . e_3)e_1 - (e_2 . e_1)e_3\|} \tag{D.1.8}$$

where $\|\ \|$, . and x are operators for vector norm, inner product and cross product, and ρ is an unknown scalar.

It is clear that e_1, e_2, and e_3 in (D.1.3)-(D.1.8) can have -1 or 1 multiplied with them, thus there are eight possible sets of solutions for G and t. However, four of them are behind the camera; that is, those solutions with $-t_3 < 0$ can be discarded. Further, if one face of the ellipse is visible while the other is not, only two of the remaining four solutions with $[0,0,-1]\ G^T t \geq 0$ are physically possible. The following procedure gives the two solutions for O and p.

One-Ellipse Locator (*ElLoc*):

1. Let $u = e_1 x\ e_2$, $v = e_3$, and $w = (e_2 . e_3)e_1 - (e_2 . e_1)e_3$
2. Let $u = u/\|u\|$, and $w = w/\|w\|$
3. If $u_3 > 0$, then $u = -u$
4. If $u . v > 0$, then $v = -v$
5. $\rho = \beta\nu/\|w\ x\ u\|$
6. $t = \rho\ u$
7. $G = [w,\ vxw,\ v]$ or $[-w, -vxw,\ v]$
8. $O = G\, Q^T$ and $p = t - G\, Q^T s$

When the model ellipse is a circle, the rotation angle around z axis is unspecified. In this case, the solutions of O and p are given by:

$$O = G \begin{bmatrix} C\theta & S\theta & 0 \\ -S\theta & C\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} Q^T \text{, and} \quad p = t - G \begin{bmatrix} C\theta & S\theta & 0 \\ -S\theta & C\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} Q^T s \qquad (D.1.9)$$

where $C\theta = \mathrm{Cos}(\theta)$ and $S\theta = \mathrm{Sin}(\theta)$. The value of θ is within $(-\pi,\pi]$. In the special case where the object is symmetric with respect to the line collinear with the z-axis of the circle, the rotation around z-axis can be ignored. However, in general, the degree of freedom in the angle of θ has to be taken into account. From (D.1.9), one has 12 linear equations for 14 unknowns: 9 in O, 3 in p, and 2 in $C\theta$ and $S\theta$. By taking an additional line or point correspondence, one has two additional linear equations in O and p. Examples for the linear equations are (D.2.1), (D.2.2), (D.3.1), and (D.3.2) in the next two sections. Thus, one can solve the undetermined angle θ, the orientation matrix O, and the position vector p by using the given circle correspondence and an additional line or point correspondence.

For one hyperbola correspondence, a similar procedure can be derived to give the two possible solutions. However, for the case of parabola, no such solution is guaranteed.

D.2 Localization Using Three Line correspondences

In the section, the problem of solving O and p by using three line correspondences is transformed into an 8-th order root-finding problem. Each real root found corresponds to a solution for O and p. Because half of the solutions are for the object behind the camera (not a visible solution), there are, at most, four possible solutions for this problem.

Let each model line, i, in the model coordinate basis be given by $\mathbf{r} = \alpha\, \mathbf{a}_i + \mathbf{r}_i$, where $\mathbf{a}_i$ is the unit directional vector of the line, α is a scalar, nd $\mathbf{r}_i$ is a point on the line. Also, let the corresponding image line be described by $Y = m_i X + T_i$, where m_i is the slope and T_i is the Y-intercept of the image line. By transforming the model lines with the orientation matrix O and positin vector p, and then perspectively projecting any two points on each transformed model line, one has the following pair of equations for each correspondence:

$$\mathbf{b}_i^T O\, \mathbf{a}_i = 0 \tag{D.2.1}$$

$$\mathbf{b}_i^T O\, \mathbf{r}_i + \mathbf{b}_i^T P = 0 \tag{D.2.2}$$

where $\mathbf{b}_i^T = [-m_i f,\ f,\ T_i]/\|[-m_i f,\ f,\ T_i]\|$, and the operator, $\|\ \|$, computes the norm of the given vector.

Equation (D.2.1) is linear in the 9 elements of the O matrix; however, to solve the 9 elements in O directly, one needs 8 line correspondences to have 8 equations like (D.2.1). The fact that matrix O has only three degrees of freedom leads to the belief that it is possible to solve for O by using only three line correspondences, though multiple solutions may exist. If the matrix O is solvable, then p can be obtained by

$$\mathbf{p} = \begin{bmatrix} \mathbf{b}_1^T \\ \mathbf{b}_2^T \\ \mathbf{b}_3^T \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{b}_1^T O\, \mathbf{r}_1 \\ \mathbf{b}_2^T O\, \mathbf{r}_2 \\ \mathbf{b}_3^T O\, \mathbf{r}_3 \end{bmatrix} , \tag{D.2.3}$$

where the first matrix is invertible if any two of the three image lines are not collinear.

The question remaining is how to solve for O by using three independent equations like (D.2.1). An approach is to eliminate two of the unknowns by using two of the three equations, thus forming an algebraic equation of degree 8 which is then solved by a root-finding technique. To do so, the first step is to find two rotation matrices F and H such that

$$F\,\mathbf{a}_1 = [1 \quad 0 \quad 0]^T \tag{D.2.4}$$

$$F\,\mathbf{a}_2 = [c_{21} \quad c_{22} \quad 0]^T \tag{D.2.5}$$

$$H^T\mathbf{b}_1 = [0 \quad 0 \quad 1]^T \tag{D.2.6}$$

$$H^T\mathbf{b}_2 = [0 \quad d_{22} \quad d_{23}]^T \tag{D.2.7}$$

where $c_{21}^2 + c_{22}^2 = 1$ and $d_{22}^2 + d_{23}^2 = 1$.

To calculate F, one can parameterize it with Roll-Pitch-Yaw angles, ϕ_z, ϕ_y, and ϕ_x, and then use (D.2.4) to solve ϕ_z and ϕ_y and use (D.2.5) to solve ϕ_x. Likewise, one can calculate H.

Once F and H are obtained, by letting $Q = H^T O F^T$ and then parameterizing the matrix Q with Roll-Pitch-Yaw angles, φ_x, φ_y, and φ_z, one has three equtions as follows:

$$[0 \quad 0 \quad 1]\; Q \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = 0 \tag{D.2.8}$$

$$[0 \quad d_{22} \quad d_{23}]\; Q \begin{bmatrix} c_{21} \\ c_{22} \\ 0 \end{bmatrix} = 0 \tag{D.2.9}$$

$$[d_{31} \, d_{32} \quad d_{33}]\; Q \begin{bmatrix} c_{31} \\ c_{32} \\ c_{33} \end{bmatrix} = 0 \tag{D.2.10}$$

where $d_3^T = [d_{31} \, d_{32} \quad d_{33}] = \mathbf{b}_3^T H$ and $c_3 = [c_{31} \, c_{32} \quad c_{33}]^T = F\,\mathbf{a}_3$

Equation (D.2.8) yields $\sin(\varphi_y) = 0$ or $\varphi_y = 0$, since pitch angle, φ_y, is in the range of $-\pi/2$ and $\pi/2$. By substituting $\varphi_y = 0$ into (D.2.9) and (D.2.10), and by defining $\mathrm{Cz} = \cos(\varphi_z)$, $\mathrm{Sz} = \sin(\varphi_z)$, $\mathrm{Cx} = \cos(\varphi_x)$ and $\mathrm{Sx} = \sin(\varphi_x)$, one obtains

$$(\alpha_1\,\mathrm{Sz}) + \mathrm{Sx}\,(\alpha_2) + \mathrm{Cx}\,(\alpha_3\,\mathrm{Cz}) = 0 \qquad \text{(D.2.11)}$$

$$(\beta_1\mathrm{Cz}+\beta_2\mathrm{Sz}) + \mathrm{Sx}(\beta_4+\beta_7\mathrm{Cz}+\beta_8\mathrm{Sz}) + \mathrm{Cx}(\beta_3+\beta_5\mathrm{Cz}+\beta_6\mathrm{Sz}) = 0 \qquad \text{(D.2.12)}$$

where $\alpha_1 = d_{22}c_{21}$, $\alpha_2 = d_{23}c_{22}$, $\alpha_3 = d_{22}c_{22}$, $\beta_1 = d_{31}c_{31}$, $\beta_2 = d_{32}c_{31}$, $\beta_3 = d_{33}c_{33}$, $\beta_4 = d_{33}c_{32}$, $\beta_5 = d_{32}c_{32}$, $\beta_6 = -d_{31}c_{32}$, $\beta_7 = -d_{32}c_{33}$, and $\beta_8 = d_{31}c_{33}$.

By replacing Cz with $(1-t^2)/(1+t^2)$ and Sz with $2t/(1+t^2)$ in (D.2.11) and (D.2.12), and by expressing Sx and Cx in terms of t, one has

$$\mathrm{Cx} = \frac{\alpha_1(\zeta_2t^2+\zeta_1t+\zeta_0)(2t) - \alpha_2(\xi_2t^2+\xi_1t+\xi_0)(1+t^2)}{\alpha_2(\eta_2t^2+\eta_1t+\eta_0)(1+t^2) - \alpha_3(\zeta_2t^2+\zeta_1t+\zeta_0)(1-t^2)} \qquad \text{(D.2.13)}$$

$$\mathrm{Sx} = \frac{\alpha_3(\xi_2t^2+\xi_1t+\xi_0)(1-t^2) - \alpha_1(\eta_2t^2+\eta_1t+\eta_0)(2t)}{\alpha_2(\eta_2t^2+\eta_1t+\eta_0)(1+t^2) - \alpha_3(\zeta_2t^2+\zeta_1t+\zeta_0)(1-t^2)} \qquad \text{(D.2.14)}$$

where $\eta_0 = \beta_3+\beta_5$, $\eta_1 = 2\beta_6$, $\eta_2 = \beta_3-\beta_5$, $\zeta_0 = \beta_4+\beta_7$, $\zeta_1 = 2\beta_8$, $\zeta_2 = \beta_4-\beta_7$, $\xi_0 = \beta_1$, $\xi_1 = 2\beta_2$, and $\xi_2 = -\beta_2$.

The fact that $\mathrm{Cx}^2 + \mathrm{Sx}^2 = 1$ yields

$$\begin{aligned}
&(\eta_2t^2+\eta_1t+\eta_0)^2[\alpha_1(2t)^2-\alpha_2(1+t^2)^2]\\
&+(\zeta_2t^2+\zeta_1t+\zeta_0)^2[\alpha_1(2t)^2-\alpha_3(1-t^2)^2]\\
&+(\xi_2t^2+\xi_1t+\text{[illegible]}\ \alpha_3\ \text{[illegible]}\ (1+t^2)^2]\\
&+2\,\alpha_2\alpha_3(1\ \text{[illegible]})(1+t^2)(\eta_2t^2+\eta_1\ \text{[illegible]})(\zeta_2t^2+\zeta_1t+\zeta_0)\\
&-2\,\alpha_1\alpha\ \text{[illegible]}t)(\text{[illegible]}t^2)(\zeta_2t^2+\zeta_1t+\zeta_0)(\text{[illegible]}^2+\xi_1t+\xi_0)\\
&-2\,\alpha\ \text{[illegible]}_3(2t)(1-t^2)(\xi_2t^2+\xi_1t+\xi_0)(\eta\ \text{[illegible]}^2+\eta_1t+\eta_0)=0
\end{aligned} \qquad \text{(D.2.15)}$$

Equation (D.2.15) is an 8-th order algebraic equation in t, which can be solved by root-finding techniques. For each root found, its correspond[illegible] and φ_x can be calculated. Subsequently, O can be found by using $O = H^TOF^T$, and [illegible] obtained by using (D.2.3). The succeeding procedure s[illegible] the steps for recoverying O and p.

Three-Line Locator (*L3Loc*):

1. Let ea[illegible]odel line be described by (a_i, r_i) and its corresponding image line by (m_i, d_i), where $i = 1,2..$
2. Find F and H such that they satisfy (D.2.4)-(D.2.7).
3. Parameterize $Q = H^T O F^T$ with Roll-Pitch-Yaw angles, φ_x, φ_y, and φ_z. Note that here φ [illegible] as inicated in (D.2.8).
4. Parameterize cos(φ [illegible]th $(1-t^2)/(1+t^2)$ and $\sin(\varphi_z)$ with $2t/(1+t^2)$.
5. Find t for (D.2.15) by using root-finding techniques.
6. For each t found in step 5, backsolve φ_z and φ_x by using theparameterization in step 4 and by using (D.2.13) and (D.2.14).Subsequently, calculate O by using the definition in step 3, and caculate p by using (D.2.3).
7. Discard any solution with $p_3 > 0$.

It interesting to note that when the first two model lines are parallel (i.e., $a_1 = a_2$), α_2 and α_3 in (D.2.11) are equal to zero; consequently, $\varphi_z = 0$ or $\varphi_z = \pi$. By expressing Cx as $(1-s^2)/(1+s^2)$ and Sx as $2s/(1+s^2)$, one has two 2nd-order equations to solve instead of the 8-th order equation in (D.2.15).

D.3. Localization Using Three Point Correspondences

The fact that n points produce n independent edge lines leads to the result that n-point localization can be done by using the n-line locating algorithm. However, for n = 3, it may be more convenient to solve the problem directly using a direct three-point locating algorithm instead of using the three-line locating algorithm.

Let the three model points be given by r_1, r_2 and r_3, and their image counterparts by (X_1,Y_1), (X_2,Y_2) and (X_3,Y_3). By projecting the model points with focal length f, one has for each correspondence, i, two eqations:

$$[f \quad 0 \quad X_i]\,(O\, r_i + p) = 0 \qquad \text{(D.3.1)}$$

$$[0 \quad f \quad Y_i]\,(O\, r_i + p) = 0 \qquad \text{(D.3.2)}$$

From (D.3.1) and (D.3.2), one concludes that $(O r_i + p)$ is proportional to the cross product of $[f \;\; 0 \;\; X_i]^T$ and $[0 \;\; f \;\; Y_i]^T$, and derives the following equations:

$$O\, r_1 + p = \sigma\, v_1 \qquad \text{(D.3.3)}$$

$$O\, r_2 + p = \rho\, v_2 \qquad \text{(D.3.4)}$$

$$O\, r_3 + p = \eta\, v_3 \qquad \text{(D.3.5)}$$

where $v_i = [X_i,\; Y_i,\; -f]$. Note that the scalars σ, ρ, and η are all positive in order to have the object in front of the camera.

Taking distance between any two of the three vectors in (D.3.3)-(D.3.5), one has

$$k_1 \equiv \|r_1 - r_2\|^2 = \eta^2\, t^T \begin{bmatrix} a_{11} & -a_{12} & 0 \\ -a_{21} & a_{22} & 0 \\ 0 & 0 & 0 \end{bmatrix} t \qquad \text{(D.3.6)}$$

$$k_2 \equiv \|r_2 - r_3\|^2 = \eta^2\, t^T \begin{bmatrix} 0 & 0 & 0 \\ 0 & a_{22} & -a_{23} \\ 0 & -a_{32} & a_{33} \end{bmatrix} t \qquad \text{(D.3.7)}$$

$$k_3 \equiv \|r_3 - r_1\|^2 = \eta^2\, t^T \begin{bmatrix} a_{11} & 0 & -a_{13} \\ 0 & 0 & 0 \\ -0_{31} & 0 & a_{33} \end{bmatrix} t \qquad \text{(D.3.8)}$$

where $a_{ij} = v_i \cdot v_j$, and $t = [\sigma/\eta,\; \rho/\eta,\; 1]$.

Moving η^2 to the left hand sides of (D.3.6)-(D.3.8) yields

$$t^T \begin{bmatrix} a_{11}/k_1 & -a_{12}/k_1 & 0 \\ -a_{21}/k_1 & a_{22}/k_1 & 0 \\ 0 & 0 & 0 \end{bmatrix} t = t^T \begin{bmatrix} 0 & 0 & 0 \\ 0 & a_{22}/k_2 & -a_{23}/k_2 \\ 0 & -a_{32}/k_2 & a_{33}/k_2 \end{bmatrix} t$$

$$= t^T \begin{bmatrix} a_{11}/k_3 & 0 & -a_{13}/k_3 \\ 0 & 0 & 0 \\ -a_{31}/k_3 & 0 & a_{33}/k_3 \end{bmatrix} t \qquad \text{(D.3.9)}$$

which then produces two conic sections:

$$t^T \begin{bmatrix} a_{11}/k_1 & -a_{12}/k_1 & 0 \\ -a_{21}/k_1 & a_{22}/k_1 - a_{22}/k_2 & a_{23}/k_2 \\ 0 & a_{32}/k_2 & -a_{33}/k_2 \end{bmatrix} t = 0 \qquad \text{(D.3.10)}$$

$$t^T \begin{bmatrix} a_{11}/k_1 - a_{11}/k_3 & -a_{12}/k_1 & a_{13}/k_3 \\ -a_{21}/k_1 & a_{22}/k_1 & 0 \\ a_{31}/k_3 & 0 & -a_{33}/k_3 \end{bmatrix} t = 0 \qquad \text{(D.3.11)}$$

where $t = [t_1, t_2, 1] = [\sigma/\eta, \rho/\eta, 1]$.

In order to solve t_1 and t_2, we must find the intersections of the two conics. One way to find the intersections of two conics is to transform first conic into its canonical form and then describe the second conic with one unknown parameter. The parameterization depends on the type of the conic in canonical form. The types are parabola, ellipse, and hyperbola. The unknown parameter in the parameterization is solved by using the second conic equation. For example, if the first conic is an ellipse with the cononical form, $(t_1/\alpha)^2 + (t_2/\beta)^2 = 1$, one can have the parameterization $t_1 = 2s/(1+s^2)\alpha$ and $t_2 = (1-s^2)/(1+s^2)\beta$ with the unknown s. The unknown parameter, s, is solved by subsitituing t_1 and t_2 into the second conic. A detail discussion of intersecting conic sections can be found in the paper by Levin (1976).

There are, at most, four intersections between the two conics in (D.3.10) and (D.3.11). Each intersection of these two conics gives a solution for t_1 and t_2. The solved t_1 and t_2 are then used to determine η via (D.3.6). Subsequently, σ and ρ can be determined. Once σ, ρ, and η are found, one has the following equations:

$$O\ (\mathbf{r}_1 - \mathbf{r}_2) = \sigma\ \mathbf{v}_1 - \rho\ \mathbf{v}_2 \tag{D.3.12}$$

$$O\ (\mathbf{r}_2 - \mathbf{r}_3) = \rho\ \mathbf{v}_2 - \eta\ \mathbf{v}_3 \tag{D.3.13}$$

To solve O and p, let $\mathbf{u}_1 = (\mathbf{r}_1 - \mathbf{r}_2)/\|\mathbf{r}_1 - \mathbf{r}_2\|$, $\mathbf{w}_1 = (\sigma\mathbf{v}_1 - \rho\mathbf{v}_2)/\|\sigma\mathbf{v}_1 - \rho\mathbf{v}_2\|$, $\mathbf{u}_2 = (\mathbf{r}_2 - \mathbf{r}_3)\ /\|\mathbf{r}_2 - \mathbf{r}_3\|$, and $\mathbf{w}_2 = (\rho\mathbf{v}_2 - \eta\mathbf{v}_3)/\|\rho\mathbf{v}_2 - \eta\mathbf{v}_3\|$. So (D.3.12) and (D.3.13) reduce to:

$$G\ \mathbf{u}_1 = F\ O\ \mathbf{u}_1 = F\ \mathbf{w}_1 = [1\ 0\ 0]^T \tag{D.3.14}$$

$$G\ \mathbf{u}_2 = F\ O\ \mathbf{u}_2 = F\ \mathbf{w}_2 = [c_{21}\ \ c_{22}\ \ 0]^T \tag{D.3.15}$$

To calculate F, one can parameterize it with Roll-Pitch-Yaw angles, ϕ_z, ϕ_y, and ϕ_x, and then use (D.3.14) to solve ϕ_z and ϕ_y and use (D.3.15) to solve ϕ_x. With the same way, one can solve for G. Consequently, one obtains $O = F^T G$. After having O, one can calculate p by using (D.3.3).

Three-point locator (*P3Loc*):

1. Let the each model point be described by $\mathbf{r}_i$ and its corresponding image point by (X_i, Y_i).
2. Let $\mathbf{t} = [\sigma/\eta,\ \rho/\eta,\ 1]^T$, where σ, ρ, and η are definded in (D.3.3)-(D.3.5).
3. Find t_1 and t_2 by intersecting the two conic sections in (D.3.10) and (D.3.11).
4. For each intersection found in step 3, find η by using

(D.3.6), and then find σ and ρ by using the definition of t in step 2.

5. Solve O by using (D.3.12)-(D.3.15) and then p by using (D.3.3).

Appendix E. Extended Kalman Filter (EKF)

Suppose the estimate of the vector **b** at time k, based on the measurements up to time instant k, is obtained as b(k|k) for the nonlinear system:

$$b(k+1) = f(b(k)) + n(k) \tag{E.1}$$

$$d(k+1) = h(b(k+1)) + v(k+1) \tag{E.2}$$

where d(k+1) is the measurement at time k+1. n(k) and v(k+1) represent unmodelled dynamics and observation noise, respectively. Both n(k) and v(k+1) are assumed to be uncorrelated and with zero-mean Gaussian distribution.

The extrapolated estimate of b(k+1), based on the information up to measurement at time k is then given by

$$b(k+1|k) = f(b(k|k)) \tag{E.3}$$

By linearizing f(b(k)) around b(k|k) and h(b(k+1)) around b(k+1|k) and by defining $e(k) \equiv b(k) - b(k|k-1)$, one has, in terms of the error e(k), the linear system:

$$\begin{aligned} e(k+1) &\equiv b(k+1) - b(k+1|k) = F(k)\ (b(k) - b(k|k)) + n(k) \\ &= F(k)\ e(k) - F(k)\ e(k|k) + n(k) \end{aligned} \tag{E.4}$$

$$g(k+1) \equiv d(k+1) - h(b(k+1|k)) = H(k+1)\ e(k+1) + v(k+1) \tag{E.5}$$

where $F(k) \equiv \partial f(b)/\partial b|_{b(k|k)}$ and $H(k+1) \equiv \partial h(b)/\partial b\ |_{b(k+1|k)}$

Let the Kalman gain (a matrix which relates the amount of update in the estimate to the difference between the measured and exrapolated attributes; its calculation is given in the procedure *DynamicEKF*) be denoted by G(k+1), then the updated estimate of e(k+1) for the linear system above is given by

$$\mathbf{e}(k+1|k+1) = \mathbf{e}(k+1|k) + G(k+1)\,[\mathbf{g}(k+1) - H(k+1)\,\mathbf{e}(k+1|k)]$$
$$= G(k+1)\,[\mathbf{d}(k+1) - \mathbf{h}(\mathbf{b}(k+1|k))] \qquad (E.6)$$

where $\mathbf{e}(k+1|k)$ is the conditional expectation of $\mathbf{e}(k+1)$ in (E.4), given measurements up to time k. By taking conditional expectation of (E.4), it is clear that $\mathbf{e}(k+1|k) = 0$.

Combining (E.4) and (E.6) yields

$$\mathbf{b}(k+1|k) = G(k+1)\,[\mathbf{d}(k+1) - \mathbf{h}(\mathbf{b}(k+1|k))] \qquad (E.7)$$

The derivation above gives the following procedure to estimate the unknown $\mathbf{b}(k+1)$ by using meaurements up to time k.

The *DynamicEKF* procedure:

```
/* The index k is for the time instant k */
1. Extrapolation of states
   $\mathbf{b}(k+1|k) = \mathbf{f}(\mathbf{b}(k|k))$
2. Extrapolation of the error covariance
   $P(k+1|k) = F(k)P(k|k)F^T(k) + Q(k)$
   where $Q(k) \equiv E\{\mathbf{n}(k)\mathbf{n}^T(k)\}$ and $F(k) \equiv \partial\mathbf{f}(\mathbf{b})/\partial\mathbf{b}|_{\mathbf{b}(k|k)}$.
3. Error covariance update
   $P(k+1|k+1) = [P^{-1}(k+1|k) + H(k+1)R^{-1}(k+1)H(k+1)]^{-1}$
   where $R(k+1) \equiv E\{\mathbf{v}(k+1)\mathbf{v}(k+1)^T\}$ and $H(k+1) \equiv \partial\mathbf{h}(\mathbf{b})/\partial\mathbf{b}|_{\mathbf{b}(k+1|k)}$.
4. Kalman Gain Matrix
   $G(k+1) = P(k+1|k+1)\ H^T(k+1)R^{-1}(k+1)$
5. State estimate update
   $\mathbf{b}(k+1|k+1) = \mathbf{b}(k+1|k) + G(k+1)\ [\mathbf{d}(k+1) - \mathbf{h}(\mathbf{b}(k+1|k))]$
```

The state estimate update can be improved by iteratively repeating Step 3 to Step 5, with correction of H(k+1) in each iteration, until no improvement can be obtained for $\mathbf{b}(k+1|k+1)$. The iteration for Step 3 to

Step 5 is done with the *ITERATE* algorimth to be given below. Details about EKF can be found in the texts of Maybeck (1982), Anderson and Moore (1977), and Balakrishnan (1984). If no dynamics exists in the given system, that is, $b(k+1) = b(k) + n(k)$, the previous *DynamicEKF* procedure is reduced to the following one.

The *StaticEKF* procedure:

1. Error covariance update

   $$P(k+1|k+1) = [[P(k|k)+Q(k)]^{-1} + H(k+1)R^{-1}(k+1)\, H(k+1)$$

   where $R(k+1) = E\{v(k+1)v(k+1)^T\}$, $Q(k) = E\{n(k)n^T(k)\}$,

   and $H(k+1) = \partial h(b)/\partial b|_{b(k|k)}$.

2. Kalman Gain Matrix

   $$G(k+1) = P(k+1|k+1)\, H^T(k+1)R^{-1}(k+1)$$

3. State estimate update

   $$b(k+1|k+1) = b(k|k) + G(k+1)\, [d(k+1) - h(b(k|k))]$$

If time index does not exist, e.g., the problem of finding a solution to minimize the averaged mean-square-error of the vector equation $d = h(b)$, the *StaticEKF* procedure is reduced to a procedure similar to the Gradient method for unconstrained optimization.

The *ITERATE* Procedure for Unconstrained Optimization:

/* the subscript, i, in this algorithm is for the i-th iteration, and should not be mistaken as time index */

1. $P_{i+1} = [P_i^{-1} + H_i R^{-1} H_i]^{-1}$

   where $R = E\{vv^T\}$ and $H_i = \partial h(b)/\partial b|_{b_i}$

2. $G_{i+1} = P_{i+1} H_i^T R^{-1}$

3. $b_{i+1} = b_i + G_{i+1}[d - h(b_i)]$

4. If $\|b_{i+1} - b_i\| \leq \epsilon$ terminate, else $i \leftarrow i+1$ and go to Step 1.

Appendix F. Camera Calibraton

This appendix summarizes an algorithm by Tsai (1986) for camera calibration. The purpose of camera calibration is to establish a model which relates the viewed point in space to the point in the image. The unknown parameters in the model of this algorithm includes camera focal length, ratio between horizotal and vertical scales, radial lens distortion parameters, and the transformation of a reference coordinate basis to the camera coordiante basis. The reference coordinate basis is the space where the reference dots are defined. The dots appear on the image, when viewed by the camera, and their positions in the image can be determined. The calibration algorithm uses the measured dot locations on the image and the known dot locations in the reference space to determine the model. Clearly, before the calibration algorithm can be used, correspondences between dots on the image and dots in the space have to be established.

Suppose the transformation between a reference coordinate basis and the camera coordinate basis is given by

$$r^c = G\, r^r + t \tag{F.1}$$

where $r^r = [x^r, y^r, z^r]^T$ is a point in the reference coordinate basis and $r^c = [x^c, y^c, z^c]^T$ is the same point in the camera coordinate basis. G and t are the rotational matrix and translational vector between the two coordinate bases.

For undistorted perspective projection with focal length f, the projected point (X_u, Y_u) on the image is given by

$$X_u = -f\, x^c/z^c \quad \text{and} \quad Y_u = -f\, y^c/z^c \tag{F.2}$$

Under radial distortion, the distorted projection (X_d, Y_d) and the undistorted projection is related by

$$X_d(1+k_1r^2+k_2r^4) = X_u \quad \text{and} \quad Y_d(1+k_1r^2+k_2r^4) = Y_u \tag{F.3}$$

where k_1 and k_2 are the radial distortion parameters, and $r = (X_d^2+Y_d^2)^{1/2}$.

From (F.2) and (F.3), it is clear that

$$X_d/Y_d = X_u/Y_u = x^c/y^c \tag{F.4}$$

Suppose the measured image point (X_m, Y_m) and the distorted projection have the relations:

$$X_m = S_xX_u \quad \text{and} \quad Y_m = S_yY_u \tag{F.5}$$

where S_x and S_y are horizontal and vertical scales.

Let $s = S_x/S_y$, $f_m = S_y f$, and $r_m = S_y r = ((X_m/s)^2+Y_m^2)^{1/2}$, one has

$$(X_m/s)(1+k_{1m}r_m^2+k_{2m}r_m^4) = -f_m x^c/z^c \tag{F.6.a}$$

$$Y_m(1+k_{1m}r_m^2+k_{2m}r_m^4) = -f_m y^c/z^c \tag{F.6.b}$$

where $k_{1m} = k_1/S_y^2$ and $k_{2m} = k_2/S_y^4$

From (F.1) and (F.6), one has G, t, f_m, s, k_{1m}, and k_{2m} as unknowns, given that X_m, Y_m, x^r, y^r, and z^r are known. The calibration process is to recover the unknowns by using the known coordinates. For convenience, it is divided into two consecutive algorithms: *CalCamera1* and *CalCamera2*.

Algorithm *CalCamera1*

Purpose: To recover G, t_x, t_y and s in (F.1) and (F.6)

1. Use (F.1), (F.4) and (F.6) to obtain for each calibration dot the equation

   $$[Y_m x^r, Y_m y^r, Y_m z^r, Y_m, -X_m x^r, -X_m y^r, -X_m z^r] \; b = X_m$$

   where $b = [t_y^{-1} s g_{11}, t_y^{-1} s g_{12}, t_y^{-1} s g_{13}, t_y^{-1} s t_x, t_y^{-1} g_{21}, t_y^{-1} g_{22}, t_y^{-1} g_{23}]^T$ and g_{ij} is (i,j) element of the G matrix.

2. Use at least 7 noncoplanar calibration points to solve the vector **b**

3. Calculate the magnitude of t_y by using the equation

   $$\|t_y\| = (b_5^2 + b_6^2 + b_7^2)^{-1/2}$$

   which results from the orthonormality of rotation matrix G and from the defination in step 1.

4. Solve s, G, and t_x using the following equations

   $$s = (b_1^2 + b_2^2 + b_3^2)^{1/2} \|t_y\|$$

   $$g_{11} = b_1 \|t_y\|/s, \quad g_{12} = b_2 \|t_y\|/s, \quad g_{13} = b_3 \|t_y\|/s,$$

   $$g_{21} = b_5 \|t_y\|, \quad g_{12} = b_6 \|t_y\|, \quad g_{13} = b_7 \|t_y\|,$$

   $$[g_{31}, g_{32}, g_{33}] = [g_{11}, g_{12}, g_{13}] \times [g_{21}, g_{22}, g_{23}]$$

   $$t_x = b_4 \|t_y\|/s,$$

5. To determine the sign of t_y, pick a point whose image (X_m, Y_m) is away from the origin. Assume, for the moment, the sign of t_y is positive, and use the obtained G, t_y and t_x to calculate x^c and y^c, as shown in (F.1). Since z^c is negative, X_m and r^c must have the same sign, so do Y_m and y^c. If this is not the case, then the sign of t_y must be negative, and the sign of the variables obtained in step 4 must be flipped.

Algorithm *CalCamera2*

Purpose: To recover f_m, t_z, k_{1m}, and k_{2m} in (F.1) and (F.6)

1. Let $w = z^c - t_z$, and compute y^c and w for each calibration point using (F.1).

2. For each calibration point, modify (F.6.b) to

$$y^c f_m + Y_m w(k_{1m} r_m^2 + k_{2m} r_m^4) + Y_m (1 + k_{1m} r_m^2 + k_{2m} r_m^4) t_z = -Y_m w \qquad (F.7)$$

2. Let $k_{1m} = k_{2m} = 0$, and, for each calibration point, reduce (F.7) to

$$y^c f_m + Y_m t_z = -w Y_m \qquad (F.8)$$

3. Use Psedo-Inverse Method to solve f_m and t_z for (F.8) of all calibration points.

4. Use f_m and t_z found in step3 along with $k_{1m} = k_{2m} = 0$ as initial guess to solve f_m, t_z, k_{1m}, and k_{2m} for the nonlinear equations, (F.7), of all calibration points.