Order Number 9034427

Machine recognition and attitude estimation of three-dimensional objects in intensity images

Gottschalk, Paul Gunther, III, Ph.D.

The University of Michigan, 1990

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

MACHINE RECOGNITION AND ATTITUDE ESTIMATION OF THREE-DIMENSIONAL OBJECTS IN INTENSITY IMAGES

by

Paul Gunther Gottschalk III

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer, Information, and Control Engineering)
in The University of Michigan
1990

Doctoral Committee:

Associate Professor Trevor Mudge, Chairman
Professor Ramesh Jain
Professor Bill Martin
Associate Professor Mike Walker
Assistant Professor Terry Weymouth

To my wife Kathy, and my daughter Elizabeth
who each showed me what is really important.

ACKNOWLEDGEMENTS

Several people contributed to the completion of this thesis. Most important was my wife Kathy. Without her love, confidence, and support you surely would not be reading this. My parents, sister Lee Ellen, brother Joel, and grandparents all helped with their love and support through the years. Special thanks go to my parents who had the forsight to have provided this thesis with an author. My advisor, Trevor Mudge, was invaluable for the support he provided. Jerry Turney deserves credit for getting me interested in object recognition, and for many insightful discussions that helped to shape this work. Amir Amini and Arnold Chiu, both graduate students working in computer vision, participated in many helpful discussions. Kunle Olukoten, Joe Dionese, Jarir Chaar, and Russell Clapp, more of my graduate student collegues, all contributed in various essential ways. The faculty and staff of the University of Michigan AI and Robotics Laboratory deserve credit for maintaining a stimulating and productive environment. Al Dobryden, Rob Giles, Dolores Bolsegna, and Virginia Folsom deserve special credit. The guys at KMS Fusion, Inc., Arnold Chiu, Jim Downward, Ted Ladewski, Fred Shebor, Jerry Turney helped at critical points, especially toward the end. I also thank the members of my committee for their suggestions and guidance.

TABLE OF CONTENTS

DEDICATION ii

ACKNOWLEDGEMENTS iii

LIST OF TABLES vii

LIST OF FIGURES viii

LIST OF APPENDICES xiv

CHAPTER

1. CHARACTERIZATION OF THE RECOGNITION PROBLEM . . . 1

1.1. Introduction to Object Recognition 1
1.2. The World Model 3
1.2.1. Modeling the 3-d Scene 3
1.2.2. Modeling the Camera 6
1.3. Some Preliminaries 8
1.3.1. Edges and Scene Discontinuities 8
1.3.2. Features and Representations 10
1.3.3. Curvature of 3-d Surfaces 11
1.3.4. Edge Contours and Contour Generators 11
1.4. The Object Recognition Task 13
1.4.1. Consequences of Employing Object-Attached Features in an Object Recognition System 15
1.4.2. The Nature and Complexity of the Object Recognition Problem 18

2. THE CYCLOPS OBJECT RECOGNITION FRAMEWORK 22

2.1. Goals of Cyclops 22
2.1.1. Sensor 22
2.1.2. Scene 23
2.1.3. Knowledge 24
2.2. Overview of Cyclops 24
2.2.1. Features 26
2.2.2. Models 26

2.2.3. Hypothesis Generation and the Model Database 29
2.2.4. Viewing Parameter Estimation 33
2.2.5. Incremental Verification 39
2.2.6. Grouping 41
2.3. Major Contributions 44

3. PREVIOUS APPROACHES TO OBJECT RECOGNITION 46

3.1. Classification of Object Recognition Methods 47
3.1.1. The Relationship Between Object Models and Sensed Data 47
3.1.2. Anatomy of Machine Recognition 48
3.2. Previous Work in Object Recognition 49
3.2.1. Transformation Clustering and Hough Methods 50
3.2.2. The Hypothesize and Verify Paradigm 60
3.2.3. Predict-Observe-Backproject Paradigm 74
3.2.4. Global Feature Methods 89
3.2.5. Optimization-Based Methods 92

4. HYPOTHESIS GENERATION AND RELATED TOPICS 96

4.1. Avoiding Object-Attached Feature Assumption: The Impact on the Hypothesis Generation Process 99
4.2. Indexing Model Views by Feature Attributes 103
4.2.1. Hypothesis Generation Viewed as Neighborhood Searches in Feature-Attribute Space 104
4.2.2. Casting Neighborhood Searching as Range Searching . . . 106
4.2.3. Efficient Implementation of Neighborhood Searches 109
4.3. Features for Hypothesis Generation in Cyclops 114
4.3.1. Properties of Good features for 3-d Object Recognition . . 115
4.3.2. Features for Hypothesis Generation in Cyclops 127
4.4. Multiview Models 149
4.5. Testing Cyclops' Hypothesis Generation Approach 154
4.5.1. Overview of the 2-d Recognition Algorithm 157
4.5.2. Feature Matching and Hypothesis Generation 160
4.5.3. Selection and Computation of Feature Vectors 160
4.5.4. Hypothesis Verification 166
4.5.5. Summary of the Algorithm and Complexity Analysis . . . 172
4.5.6. Experimental Results 175
4.5.7. Conclusion 183
4.6. Chapter Summary 185

5. MODEL-BASED VIEWING PARAMETER ESTIMATION AND TRACKING 187

5.1. Overview of Attitude Estimation by Feature Modulated Attractors (AEFMA) 189
5.1.1. Pseudo-Code Description of AEFMA 195

5.1.2. Continuous Versus Discrete Optimization 198
5.2. Design of the Interfeature Disparity Function 202
5.3. 2-d Shape Representation for Viewing Parameter Estimation 210
5.3.1. Selectiveness in the Context of Shape primitives 213
5.3.2. Attribute Smoothness in the Context of Shape primitives . 214
5.3.3. Attribute Invariance in the Context of Shape primitives . . 215
5.3.4. Computation of the Shape primitives 217
5.3.5. Sparse versus Complete Shape Representation 218
5.3.6. Computation of the Shape Representation 224
5.4. Computing the Composite Disparity Function 226
5.4.1. Computing the Composite Disparity Function Efficiently . 226
5.4.2. Weighting the Shape primitives 228
5.5. Minimizing the Composite Disparity Function 229
5.5.1. Viewpoint Parameterization 230
5.5.2. Empirical Observations on the Nature of the CDF 232
5.5.3. Optimization 234
5.6. AEFMA's Relationship to Previous Work 238
5.6.1. Methods that use Object-Attached Shape Representations and AEFMA 239
5.6.2. Related Methods 241
5.7. Human Attitude Estimation 245
5.8. Models 247
5.9. Experiments in Attitude Estimation and Tracking with AEFMA . . 251
5.9.1. Methods 251
5.9.2. Results 253

6. CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS 286

6.1. Summary of Contributions 286
6.2. Directions for Future Research 288

APPENDICES 289

A.1. Overview of the Knowledge-Based Contour Grouping Module . . 290
A.2. Hot Spot Detection 292
A.3. Knowledge-Based Contour Grouping 292
A.3.1. Global Data Structure 293
A.3.2. Fuzzy Logic 299
A.3.3. Rules and Control 300
B.1. Results of Experiments on Images of Overlapping Puzzle Pieces . 306
B.2. Results of Experiments on Images of Overlapping Parts of a Switch Assembly 310

BIBLIOGRAPHY 315

LIST OF TABLES

Table

1.1 Object Attached Features 15

4.1 Algorithms for range searching 111

4.2 Results of Contour Grouping Module 151

4.3 Summary of False Alarm Statistics for 2-d Experiments 179

5.1 Factors influencing the design of the IDF. 203

LIST OF FIGURES

Figure		
1.1	Coordinate Frames	4
1.2	Perspective Projection	5
1.3	Orthographic Projection	7
1.4	Surface Curvature	12
1.5	Silhouette Generator	12
1.6	Occluding Contours are not Object-Attached	16
2.1	Overview of Cyclops	25
2.2	Multiview Model	27
2.3	A feature employed by Cyclops	31
2.4	How a k-d tree indexes feature space	32
2.5	AEFMA approach	35
2.6	A potential function with a singularity	37
2.7	A 2-d Gaussian IDF	38
2.8	Result of Running AEFMA	39
2.9	Local grouping of three contours	44
3.1	A vertex pair	53
3.2	Affine invariance of Lamdan *et al*'s features	57
3.3	Noiseless Backprojection	77
3.4	Noisy Backprojection	78

3.5 Approximation to full backprojection 79
3.6 Visibility Locus 83
4.1 Topics of Chapter 4 97
4.2 Projection of Cube Corners 101
4.3 Viewpoint Dependency of High Curvature Points 102
4.4 A Neighborhood Query 105
4.5 The effect of changing the size of a neighborhood query 106
4.6 Geometric View of Range Queries 107
4.7 How a k-d tree indexes feature space 112
4.8 Queries on a k-d tree 113
4.9 An Example k-d tree 114
4.10 Selectiveness of 2-d Doorlock Part Segments 119
4.11 Attneave's Cat 120
4.12 Lowe's Cat 121
4.13 Point Pair Features 129
4.14 An Example Image 132
4.15 Result of Canny's edge detector 133
4.16 Result of Linking edges 134
4.17 Calculating a Uniform Arclength Parameterization 135
4.18 An Example of the dual (x, y, θ)-s Representation 136
4.19 Result of Multiscale Critical Point Detection 138
4.20 Detection of Significant Inflection Points 140
4.21 Attribute Symmetries in Point-Pair Features 143
4.22 Contour Relations Considered by the Contour Grouping Module . . . 146
4.23 Indirect Edge Contour Grouping Evidence. 147

4.24 An Image of a Complex Scene 148
4.25 Hots pots Detected in Complex Scene 149
4.26 ID Numbers of Contours 150
4.27 Multiview Model 152
4.28 Result of uniform viewpoint algorithm using six points 155
4.29 Result of running uniform viewpoint algorithm using 101 points . . . 156
4.30 Steps in the 2-d Recognition Algorithm 159
4.31 CPN features in Cartesian and θ-s representations 161
4.32 Geometry of K-L expansion 163
4.33 Basis Vectors 165
4.34 CPN features 166
4.35 Constructing a 2-d Hypothesis 168
4.36 Calculating the fraction of boundary matched 171
4.37 Hypothesis Acceptance Region 173
4.38 Pseudocode recognition algorithm. 174
4.39 Models of the Jigsaw Puzzle Pieces 177
4.40 Models of the Switch Parts 178
4.41 Results for Image of Overlapping Puzzle Pieces 181
4.42 Recognized Puzzle Pieces vs. Fraction of Boundary Visible 182
4.43 Results for Image of Overlapping Switch Parts 183
4.44 Switch Parts Recognized vs. Fraction of Boundary Visible 183
5.1 AEFMA's place in Cyclops. 189
5.2 Model-based tracking using AEFMA 190
5.3 Tracking subproblems 191
5.4 AEFMA approach 193

5.5 Conceptual pseudocode description of the CDF 196
5.6 Pseudocode for computing the CDF efficiently. 197
5.7 Top level pseudocode of AEFMA. 199
5.8 Effect of reducing the size of the components of the σ-vector. 200
5.9 2-d candidate for the IDF with a curvature singularity. 203
5.10 2-d symmetric Gaussian candidate for the IDF. 204
5.11 Summation of symmetrically placed Gaussian IDF's 206
5.12 Summation of symmetrically placed IDF's containing singularities . . 207
5.13 A shape primitive. 211
5.14 Empirical demonstration of the selectiveness of shape primitives. . . . 215
5.15 Empirical demonstration of the selectiveness of shape primitives. . . . 216
5.16 Empirical demonstration of the selectiveness of shape primitives. . . . 216
5.17 Attribute symmetries in a shape primitive. 219
5.18 Misalignment of shape primitives resulting from sparse placement. . . 220
5.19 Marker points allow sparse placment of primary points 221
5.20 Barriers in the CDF that can result from use of marker points. 223
5.21 A 2-d slice of an actual CDF using marker points. 224
5.22 Effect of moving primary points close together 225
5.23 Parameterization of the Out-of-Plane Rotations. 231
5.24 A 2-d slice of an actual CDF 234
5.25 The translational bounds used in Figs. 5.24, 5.26, and 5.27. 235
5.26 A 2-d slice of an actual CDF 236
5.27 A 2-d slice of an actual CDF 237
5.28 Pose estimation under object-attached feature assumption. 240
5.29 Stimuli used in Tarr and Pinker's work. 246

5.30 Van Hove's representation of the surface of a model. 249
5.31 Essence of Basri and Ullman's approach to edge contour prediction. . 250
5.32 Predicting edge contours of the space shuttle. 255
5.33 An image of the space shuttle 256
5.34 Some of the parts used in other experiments with AEFMA. 257
5.35 Run one testing the convergence of AEFMA. 258
5.36 Run two testing the convergence of AEFMA. 259
5.37 Run three testing the convergence of AEFMA. 260
5.38 Run four testing the convergence of AEFMA. 261
5.39 Run five testing the convergence of AEFMA. 262
5.40 Run six testing the convergence of AEFMA. 263
5.41 Tracking using AEFMA: frame one. 264
5.42 Tracking using AEFMA: frame two. 265
5.43 Tracking using AEFMA: frame three. 266
5.44 Tracking using AEFMA: frame four. 267
5.45 Tracking using AEFMA: frame five. 268
5.46 Docking experiment using AEFMA: frame one. 270
5.47 Docking experiment using AEFMA: frame two. 271
5.48 Docking experiment using AEFMA: frame three. 272
5.49 Docking experiment using AEFMA: frame four. 273
5.50 A Tilted shuttle. 275
5.51 Another tilted shuttle 276
5.52 Tilted shuttle in a cluttered image. 277
5.53 Tracking rotations out the image plane: frame one of four 279
5.54 Tracking rotations out of the image plane: frame two of four 280

5.55 Tracking rotations out of the image plane: frame three of four 281
5.56 Tracking rotations out of the image plane: frame four of four 282
5.57 Multiple similar objects with oblong model. 284
5.58 Multiple similar objects with oblong model. 285
A.1 Contour Data Structure 293
A.2 Format of the contour connection connection list 294
A.3 Connection label data structure 294
A.4 Example of a contour data structure 295
A.5 Fuzzy conversion functions 300
A.6 An example rule 301
A.7 An example rule 304
A.8 Another example rule 304
B.1 Result of Recognition on an Image of Overlapping Puzzle Pieces . . . 307
B.2 Result of Recognition on an Image of Overlapping Puzzle Pieces . . . 307
B.3 Result of Recognition on an Image of Overlapping Puzzle Pieces . . . 308
B.4 Result of Recognition on an Image of Overlapping Puzzle Pieces . . . 308
B.5 Result of Recognition on an Image of Overlapping Puzzle Pieces . . . 309
B.6 Result of Recognition on an Image of Overlapping Switch Parts . . . 311
B.7 Result of Recognition on an Image of Overlapping Switch Parts . . . 311
B.8 Result of Recognition on an Image of Overlapping Switch Parts . . . 312
B.9 Result of Recognition on an Image of Overlapping Switch Parts . . . 312
B.10 Result of Recognition on an Image of Overlapping Switch Parts . . . 313
B.11 Result of Recognition on an Image of Overlapping Switch Parts . . . 313
B.12 Result of Recognition on an Image of Overlapping Switch Parts . . . 314
B.13 Result of Recognition on an Image of Overlapping Switch Parts . . . 314

LIST OF APPENDICES

Appendix

A. KNOWLEDGE-BASED CONTOUR GROUPING 290

B. ADDITIONAL 2-D RECOGNITION RESULTS 306

CHAPTER 1

CHARACTERIZATION OF THE RECOGNITION PROBLEM

1.1 Introduction to Object Recognition

Visual recognition is the process of explaining what is sensed with particular instances of models of objects. We will call this set of models the system's *vocabulary* of objects. This thesis addresses the automatic recognition and attitude determination of three-dimensional (3-d) objects in single two-dimensional (2-d) intensity images using only information about the *shape* of the objects. Biological vision systems employ many cues to assist in segmenting and recognizing objects; shape, texture, color, shading, and specularity are a few of them. Unfortunately, the state of the art in computer vision is not sufficiently advanced to effectively utilize multiple sources of information to recognize objects. Indeed, much work remains to be done on the more basic problem of how to extract such cues, much less about using them for recognition. As a subgoal, shape based recognition has two advantages: first, shape is among the better understood cues mentioned above; and second, there are many practical applications for a shape based recognition. The second point is particularly true when the objects to be recognized are of man-made origin, as is the case in many industrial settings as well as in space. In such circumstances, objects to be recognized are often manufactured via the same process using identical materials, and therefore cannot be distinguished on the basis of surface characteristics. Thus, shape is the primary cue for distinguishing such objects.

Therefore, while it is clear that shape-based recognition is only a piece in the puzzle of object recognition, few would argue that it is an important, fascinating, and useful part of it.

In addition to its interesting and challenging nature, shape-based recognition of 3-d objects in 2-d intensity images deserves study for three reasons:

- a general solution would have a wide practical application,
- sufficiently general solutions have not yet been found, and
- compared to other sensors, high-resolution intensity sensors are cheap, rugged, and readily available.

There are many modes of sensing now available. In particular, there are the means directly measure the scene geometry to obtain *range images*, either via special purpose sensors, or via indirect methods such as stereo or structured light. For a number of reasons discussed later, recognition from range data is simpler than recognition from intensity data. If this is so, why study recognition using intensity data if range data is available? Active range sensors are larger, more expensive, and more fragile, have smaller data bandwidth, and often poorer quality data than conventional cameras. Indirect methods only yield accurate range points at sparsely spaced points, implying that some type of surface interpolation must be done. In addition to being time consuming, the surface interpolation problem is far from being solved. Perhaps most persuasive, however, is the fact that human and animal eyes are intensity sensors. That they can recognize objects easily and quickly is a challenge that is hard to ignore.

This thesis describes a framework for recognition of 3-d objects in intesity images called Cyclops. Cyclops approaches 3-d recognition in a novel manner that allows it to handle *any* rigid object. Further, the method is extensible to articulated and deformable objects. This is one way in which Cyclops is an improvement over previous algorithms, as previous algorithms have had severe restrictions on their vocabulary. For example, many existing systems require that the objects in their vocabularies be polyhedral or planar.

The cost for Cyclops's generality is added complexity. The reasons for this are of a fundamental nature and will occupy us many times thoughout the remainder of this thesis. However, before proceeding, it will be necessary to dispense with some preliminaries.

1.2 The World Model

In order to recover 3-d information from a 2-d image, an object recognition system must have a model of the environment. The modeling task consists of two major parts: modeling the 3-d world and modeling the imaging of the 3-d world on the 2-d image plane. The 3-d world can be modeled as a subpart hierarchy, with transformations among the world frame and the subparts determining the geometrical configuration of the world. An important part of recognition is estimating the transformations between the world and its subparts. Thus, it is necessary to choose the natural coordinate frames for recognition. Further, we must determine which transformations the system must compute and which are given. Finally, we must model the imaging process.

1.2.1 Modeling the 3-d Scene

In rigid object recognition tasks, there are typically five coordinate frames that are of interest: the *scene* (or *world*) frame, the *viewer* (or *camera*) frame, the *model* (or *object*) frame, the *image* frame, and the *sensor* frame. Fig. 1.1 shows the relationships between the various coordinate frames. The scene frame is the global frame in which the location of recognized objects are reported to the user. The origin of an object (or model) frame is usually attached to a specific (often arbitrary) reference point within an object (or model) and stays fixed relative to all points in the object (or model). The camera frame is similar to an object frame except it is fixed relative to the camera. The details of the placement of the camera frame are shown in Fig. 1.2, and are discussed in the following section.

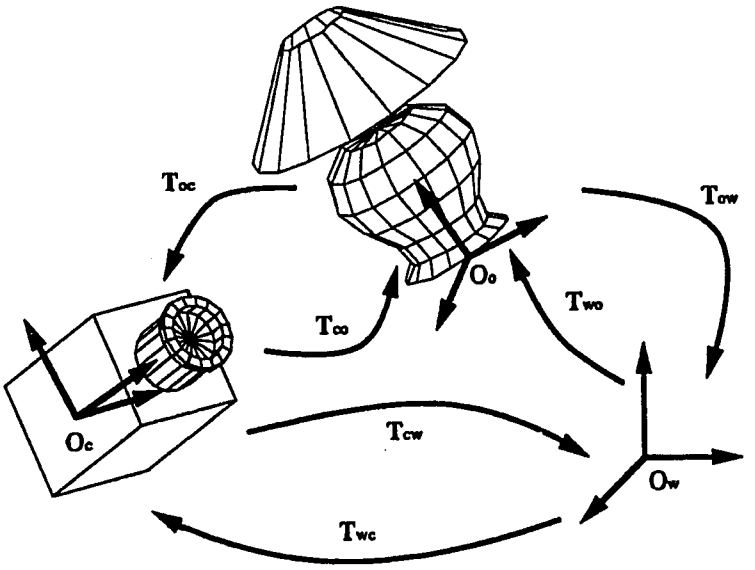


Figure 1.1. The relationship between coordinate frames in the 3-d world model. O_o is the origin of the object frame, O_c is the origin of the camera frame, and O_w is the origin of the world frame. Also shown are the transformations between the frames.

The image frame and the sensor frame differ from the other frames in that they are 2-d frames. Both frames index spatial locations within the image plane. Thus, only one of them is strictly necessary. The sensor frame, however, is often inconvenient to work with since it is usually chosen (arbitrarily for our purpose) to allow each pixel in the image to be indexed by integers. In addition, the sensor frame is usually left handed. Therefore, it is useful to introduce the *image* coordinate frame. The image frame is right-handed and has its origin at the center of the image (rectangular images are assumed). In this thesis, the scaling of the axes is chosen to just fit the image inside the square $\{(x, y) \in [-1, 1]\}$, though any convenient scale could be used.

Identification of the important coordinate frames leads us to consider which transformations between them are known *a priori*, and which the object recognition system is expected to determine. If $\mathbf{p}_{src}$ denotes the coordinates of p in the source frame, *src*, and $\mathbf{p}_{dst}$ denotes the coordinates of p in the destination frame, *dst*, then $\mathbf{p}_{dst} = T_{dst\ src}\mathbf{p}_{src}$ denotes the transformation from the coordinates of p in frame *src* to frame *dst*. When rec-

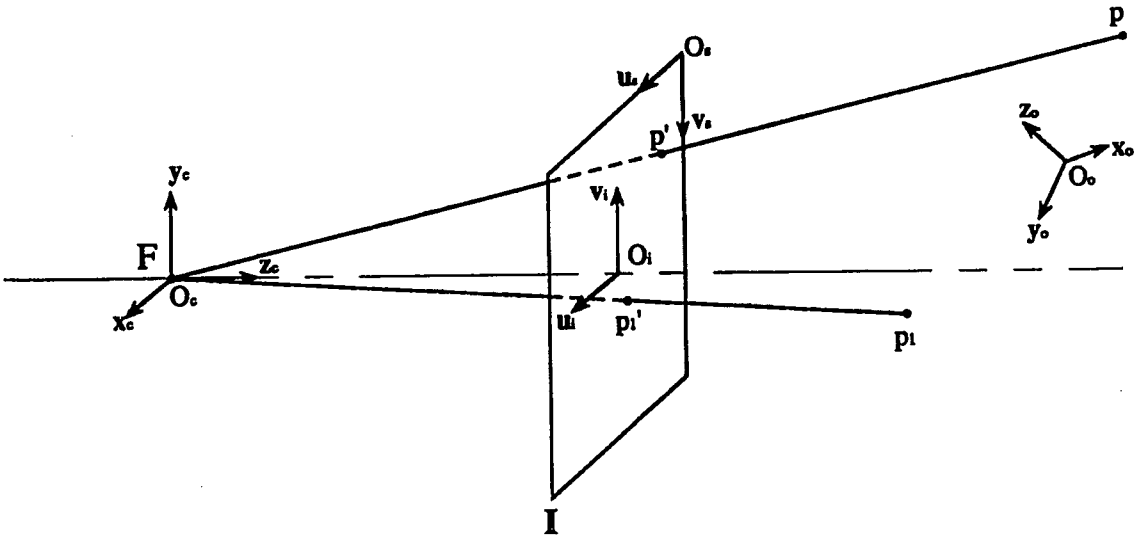


Figure 1.2. The geometry of perspective projection. F is the focal point, and I is the image plane. O_o is the origin of the object frame, O_c is the origin of the camera frame, O_i is the origin of the image frame, and O_s is the origin of the sensor frame. The 3-d scene points p and p_1 project to the 2-d points p' and p_1' in the image plane.

ognizing 3-d objects from 2-d images, we are given an image and T_{wc}, where w denotes the world frame, and c the camera frame. When an object is recognized, the user desires the position and orientation of the object (also referred to as *pose*, *attitude*, or *viewing parameters*). This is equivalent to knowing T_{wo}, where w is as before and o denotes the object frame. Most recognition systems determine T_{co} and compute T_{wo} by composing T_{wc} and T_{co}. The process of finding T_{wo} is referred to in several ways, among them *pose determination*, *attitude estimation* and *viewing parameters estimation*. Viewing parameters specify the information that is necessary to produce a graphical rendering of the object, and therefore can include not only T_{co}, but photometric information such as the surface reflectance of the objects and lighting of the scene, and camera parameters such as the focal length. In this thesis, the viewing parameters will specify only T_{co} and the camera parameters. Lighting parameters are largely ignorable because we use edge-based features and image representations that possess a large measure of insensitivity to lighting variations.

1.2.2 Modeling the Camera

Ideally, a point in the scene project to a point in the image. Such behavior is theoretically attainable only in a perfect pinhole camera. In such an ideal system, the laws of ray optics reign, and using simple geometry it is possible to determine the image coordinates (u^p, v^p) of the projection of any scene point p located in the world coordinate system at (x_w^p, y_w^p, z_w^p). Figure 1.2 shows the geometry of *perspective projection*, the most accurate model of projection for the pinhole camera. The focal point f is located at $(0, 0, 0)$ in the camera body coordinate system, whose origin in the figure is at O_c, and whose axes are denoted by x_c, y_c, and z_c. The image plane is the plane $z = f$. Given point p with camera body coordinates (x_c^p, y_c^p, z_c^p), it is simple to show that the coordinates of p', the projection of p, are

$$u^p = \frac{f x_c^p}{z_c^p}$$

$$v^p = \frac{f x_c^p}{z_c^p} \tag{1.1}$$

in the image frame.

In contrast to pinhole cameras, practical cameras employ lens systems in order to gather sufficient light. The additional light is not without cost, however. The large aperture of a lens leads to a limited range of distances (called the *depth of field*, determined by the f-number of the lens system) from the camera where a point in the scene projects adequately to a point in the image; outside this range, a point projects to a *blur circle*. Indeed, there is only one plane in the scene, the focal plane, which is parallel to the image plane for a paraxial lens system, whose points project to points in the image plane. However, this is true only for a perfect lens under the assumption of ray optics. In realistic cameras, lens imperfections such as chromatic aberration, spherical aberration, coma, and astigmatism blur their projections. Fortunately, the perspective model of projection holds quite well even for realistic lens imaging systems

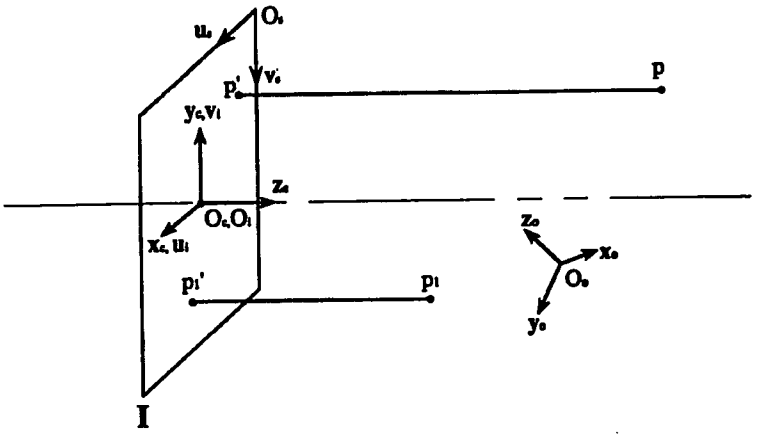


Figure 1.3. The geometry of orthographic projection. The coordinate frames are as in Fig. 1.2

An accurate model of the imaging process takes the effects mentioned in the previous paragraph into account. For the purposes of this thesis, it is not necessary use a model that includes these effects. In particular, the camera will be modeled as a pinhole camera resulting in infinite depth of field. Any deviations from this ideal behavior will be modeled as sources of noise and distortion that the recognition system must contend with.

In many situations in computer vision, particularly those involving inverse problems, i.e., those that involve solving for the viewing parameters given a set of 3-d model points and their corresponding 2-d projections, the analysis is simplified considerably if full perspective is approximated by *weak perspective* [TM87, Hut88]. Weak perspective approximates full perspective as a scaled orthographic projection. In full perspective scene points are connected to their projections in the image plane by rays passing through themselves, their projections and the focal point, as shown in Figure 1.2. In contrast, orthographically projected points are connected to their projections by rays that are perpendicular to the image plane and that pass through both them and their projections, as shown in Fig. 1.3.

Weak perspective holds when the dimensions of an object are small compared to the distance from the viewer to the object. Let **p** be the vector from the origin of the camera

coordinate system to a point p on an object. The vector $\mathbf{p}$ can be written as $\mathbf{p} = \mathbf{c} + \mathbf{r}$, where $\mathbf{c}$ is the location of a reference point within the object to which p belongs, and $\mathbf{r}$ is the location of p relative to the reference point. Let the coordinates of p in the camera frame be (x^p, y^p, z^p), and the coordinates of C be (x^c, y^c, z^c). Then the image coordinates $(u^{p'}, v^{p'})$ of p', the projection of p are given by

$$u^{p'} = \frac{fx}{z^c}$$
$$v^{p'} = \frac{fy}{z^c} \tag{1.2}$$

where f is the focal distance. If we let $s = f/z^c$, then we have $u^{p'} = sx$ and $v^{p'} = sy$, a simple scaling of the orthographic projection of p.

1.3 Some Preliminaries

This section introduces terms and concepts that may be unfamiliar to readers not intimately familiar with computer vision, and defines, for the context of this thesis, terms that have several usages in the literature.

1.3.1 Edges and Scene Discontinuities

A discontinuity in any measurable image property can be termed an edge. Most commonly, edges are discontinuities in the local intensity measured by the camera, although texture and range edges have also been studied. In this thesis, *edge* will always refer to an intensity discontinuity.

Edges in the image carry much information as they often demarcate the boundaries of objects. Also, the situations in a scene that can give rise to an edge are few and relatively well defined. Further, edges are relatively insensitive to variations in lighting. For these reasons edge-based object recognition methods have flourished more than methods that are based on region segmentation.

While there are a number of mechanisms for generating edges in images, let us restrict our attention for the moment to 3-d surface geometries that tend to generate edges. There are two cases:

- Occluding boundaries.
- 3-d tangent discontinuities.

An *occluding boundary* is formed when one surface obscures another from view, leading to a discontinuity in depth along the line of sight. Such surfaces belong to the same object. If so, we call it *self-occlusion.*

Occlusion boundaries tend to generate edges in images under widely varying scene conditions. In this respect such edges are very robust features. The reason for this is the fact that the occluding surface is likely to differ significantly from the back surface in one or more of the following three ways:

- surface composition,
- orientation, and
- illumination.

A discontinuity in any of these surface properties is likely to lead to a discontinuity in the intensity of light reflected toward the camera as the boundary is crossed.

The other type of 3-d geometry that can generate edges is the *3-d tangent discontinuity*, also referred to as a *crease.* A crease is a discontinuity of the normal to a surface. Creases are much less likely to reliably generate an edge under a wide range viewing conditions than occluding boundaries. This is because, only surface orientation is likely to be discontinuous across a crease. Surface reflectance is likely to be identical; since the surfaces are continuous, it very likely that they are from the same object, and, therefore, possess identical reflectence propterties. Similarly, the illumination is likely to be continuous across the crease.

In real scenes, true 3-d tangent discontinuities do not exist: at some scale the surface can be viewed as continuous. Since all edge detectors have some decision threshold built

into them, edges may therefore be generated by highly curving 3-d surfaces as well, with still less reliability.

1.3.2 Features and Representations

Examination of the literature shows that little agreement exists on what exactly a feature is, or is not. Generally, they all agree that a feature is an abstraction of the image data that expedites the recognition task. Typically, generating a feature involves two types of processes: grouping processes and detection processes. A good low-level example is the intensely scrutinized problem of finding edge contours from an image. This problem is usually decomposed into "edge detection" followed by "edge linking". Edge detection is the process of finding plausible candidate edge points, and, as indicated by its name, is a detection process. The linking process groups the candidate pixels into contiguous chains. The resulting chains, or contours, are features.

Although we have called edge contours features, many would say that they are really an example of a *shape representation.* We do not argue with this; rather, we believe that there is little or no difference between what is a feature and what is a representation. It appears that the more complete the description of the shape (or other properties) of the object, the more likely it is to be called a representation, while sparse descriptions, especially those that preserve relatively little of the available shape information, tend to be called features. For example, while the edge contours are likely to be called a representation, circular arcs, line segments, and "corners" are usually referred to as features. We will treat less complete shape descriptions as *features*, and will tend to call more complete shape descriptions *shape representations.*

A feature can always be represented by a vector of *attributes.* In general, each such attribute may be continuous, as in the case of the *eccentricity* attribute of an ellipse feature, or discrete, as in the case of the possible values *acute*, *right*, and *obtuse* of the

angle attribute of a **junction** feature. A vector of attributes will be referred to as a *feature vector.*

1.3.3 Curvature of 3-d Surfaces

The study of the differential properties of 3-d surfaces is the realm of differential geometry. One of the most important means of characterizing the differential properties of a surface is through curvature. Given a surface S, a point on it P, and a normal to the surface $\mathbf{n}$, as shown in Figure 1.4, then we can define the curvature in the direction $\vec{v}$ in the tangent plane at p as the curvature of the space curve C that is obtained from the intersection of the plane containing both $\mathbf{n}$ and $\mathbf{v}$ at p. Clearly, as $\mathbf{v}$ rotates in the tangent plane, the curvature of the surface in the direction of $\mathbf{v}$ changes and is a periodic function of the rotational angle. For some direction $\mathbf{v}_{min}$ the curvature takes on its smallest value, ρ_{min} while at some other direction $\mathbf{v}_{max}$ the curvature takes on its largest value, ρ_{max}. The directions $\mathbf{v}_{max}$ and $\mathbf{v}_{min}$ are called the principle directions on the surface at p, and the corresponding curvatures ρ_{min} and ρ_{max} are called the principle curvatures. It can be shown that the principle directions are always orthogonal [One66]. In addition to the principle curvatures, other curvature measures can be defined, such as *Gaussian curvature* which is the product of the principle curvatures and *mean curvature* which is the average of the principle curvatures.

1.3.4 Edge Contours and Contour Generators

Assuming that objects are opague, the scene point that projects to a particular point in the image is the frontmost surface intersected by the line of sight. Thus, for any set of image points, there is a corresponding set of scene points that projects to the image set. If the image set happens to be an edge contour, the locus of scene points that project to it is called a *contour generator.* One very interesting type of contour generator is a

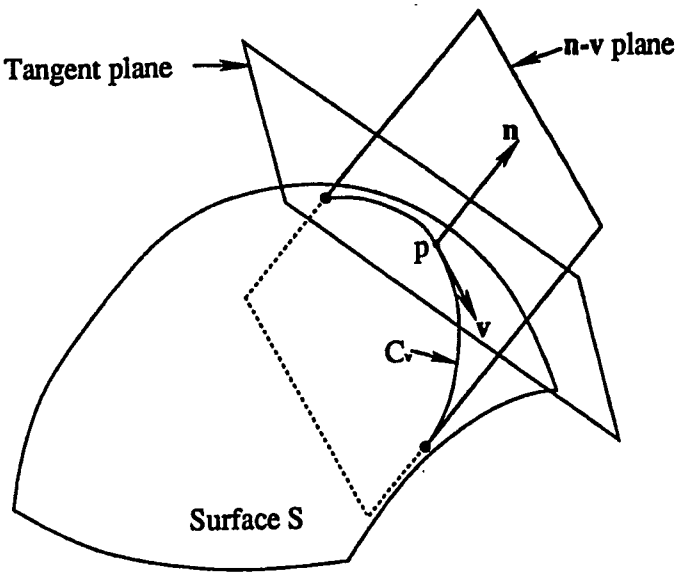


Figure 1.4. The differential geometry of a surface defining directional curvatures.

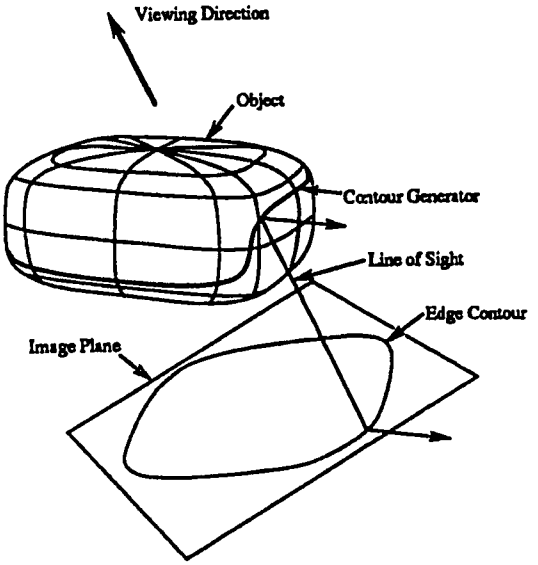


Figure 1.5. The one type of contour generator is the set of points on the object that are normal to the lines of sight. The resulting edge contour in this case is an *occluding boundary*

smoothly curving occluding boundary. Such a contour generator is shown in Fig. 1.5. As we will see shortly, such contours are an example of a non-object-attached feature.

1.4 The Object Recognition Task

The information usually required of an object recognition system consists of a list of *model instances*, consisting of pairs (O, T_{wo}), where O is an object identifier and T_{wo} is the system's estimate of the transformation from object to world coordinates. Each entry indicates that the recognition system has enough evidence to believe that the object is present in the scene. This leads to the definition of object recognition that we will use:

> Object recognition is the process of finding a set of model instances that predict well what is observed in the image.

This definition automatically embodies what Lowe has called "the viewpoint consistency constraint" [Low87b]:

> "The locations of all projected model features in an image must be consistent with projection from a single viewpoint."

In the 3-d recognition task considered here, objects may appear in scenes in any orientation relative to eachother. This implies that global, inter-object viewpoint consistency is of little use here. However, the features that are matched to a particular object model must be consistent with projection of that model from a single viewpoint. This is intra-object viewpoint consistency, and is automatically part of the definition of object recognition given above, by virtue of the fact that a model instance is a model transformed to a particular unique viewpoint.

Unfortunately, there is no widely accepted definition of object recognition. Many researchers make up one to suit themselves. However, many of them have the flavor of the following two examples. The first, taken from the recent work of Huttenlocher [Hut88], is:

> "Recognizing an instance of an object involves finding a consistent set of corresponding model and image features, such that some transformation maps each model feature onto its corresponding image feature."

The second, taken from the early work of Roberts [Rob64], is:

> "... R transforms a model into an object and P transforms the object onto the picture so that if $H = RP$, H transforms the model points to picture points. Therefore, in order to identify a group of points and lines in the picture with a particular model, we must find out if there is any transformation H which will take the model's points and lines into those of the picture. If such a model and transform are found, it can be said that the object represented in the picture could be that model under the transform $R = HP^{-1}$."

Implicit in both of these definitions, as in most others, is the idea that there exists a *correspondence* between model features, which are 3-d entities, and image features, which are 2-d entities. Further, it is implicit that when these 3-d entities are projected, they reliably and consistently yield 2-d entities, or features, that are easily recognizable. The assumption is usually made that the 3-d entities (usually called *3-d features*) will reliably project to these special, easily segmented 2-d features under all viewpoints where the 3-d feature is visible. We will refer to such features *object-attached* features, since such features behave as though they were glued to the surface of an object and then projected. In other words, 2-d object-attached feature arises from the projection of a very specific portion of the surface of the object comprising the 3-d feature, regardless of the viewpoint. Features that do not have this property will be called either *general* or *non-object-attached* features. When recognition or attitude estimation systems rely on the properties of object-attached features, we will say they are invoking the *object-attached feature assumption.* We will show shortly that invoking this assumption results in severe restrictions on the types of objects that a system can handle.

The impact of object-attached features on solving the recognition task has been largely ignored. In Chapter 3 we observe that the vast majority of recognition methods, particularly those using spatially local features, invoke the object-attached feature assumption. It is remarkable that the object-attached feature assumption is so prevalent, but to our

Principle Curvatures	$\rho_2 \approx \infty$		
	$\rho_1 \approx 0$ at a point	$\rho_1 \approx 0$ in a neighborhood	$\rho_1 \approx \infty$
Example	Inflection point in the occluding boundary of a planar object.	Intersection of two planar surfaces.	Apex of a cone

Table 1.1. 3-d structures generating object-attached features.

knowledge has rarely been acknowledged. To our knowledge, only Nalwa [Nal88], in a paper on the representation of smooth 3-d surfaces, has explicitly drawn this distinction between the two feature types. He calls them "viewpoint independent" and "viewpoint dependent" for object-attached and non-object-attached respectively. The apparent incognizance of the significance of the object-attached feature assumption is still more suprising considering the simplifications in the recognition and attitude determination processes.

1.4.1 Consequences of Employing Object-Attached Features in an Object Recognition System

At this point, the reader may be wondering if there are features that are not object-attached, indeed, aren't all features projections of a particular 3-d surface structure? In fact, most features are *not* object-attached. Further, the list of 3-d features that can be reliably assumed to generate object-attached features is very short. They can be enumerated in terms of the principle curvatures, ρ_1 and ρ_2, in by Table 1.1.

Table 1.1 lists all of the 3-d surface configurations that can lead to 2-d features that can be reliably considered object-attached is small and specialized. Objects that have few or none of these structures cannot be recognized by systems that employ the object-attached feature assumption. A particularly important example of a situation where a feature cannot be considered object-attached is an occluding boundary at points where the surface of the object has finite curvature in the direction normal to the contour

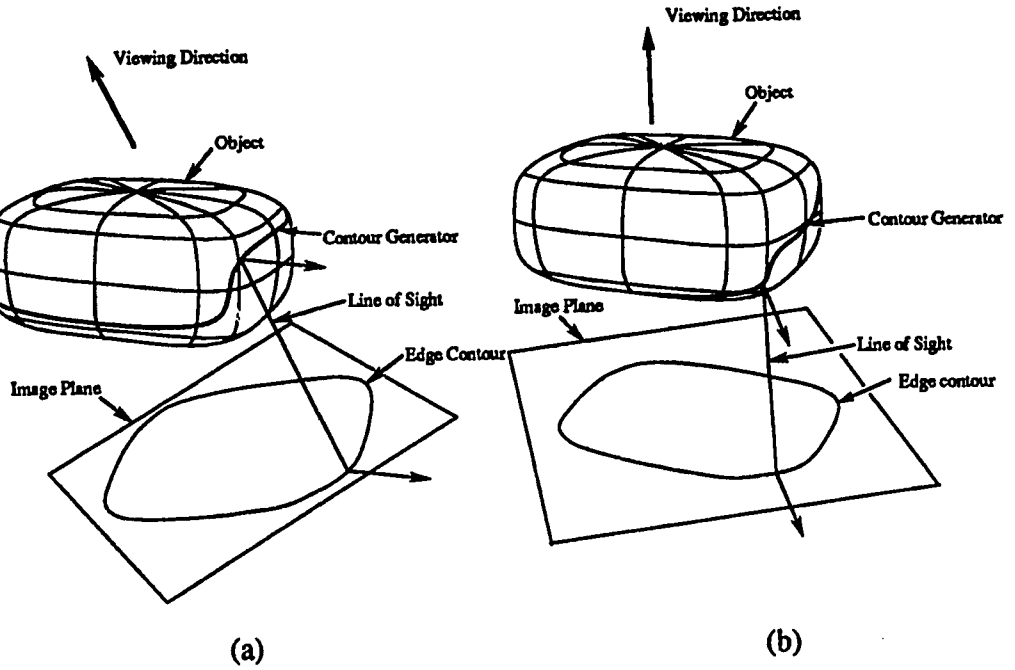


Figure 1.6. As the viewing direction is changed, as from (a) to (b), the points on the surface of a smooth object comprising the contour generator change position withing the surface of the object, implying that the such edge contours are *not* object-attached features.

generator. In this situation, changing the viewpoint changes the position contour generator within the surface of the object. Therefore, the edges generated by the contour generator are *not* object-attached features. Fig. 1.6 demonstrates this fact graphically. Hence, any system that relies on the object-attached feature assumption cannot use the edges that are projections of a smoothly curving occluding contour This is unfortunate since occluding surfaces, as we pointed out earlier, are very robust features. In addition, the fact that people usually have no trouble recognizing familiar objects from partial occluding contours implies that contain much information about the shape of an object.

As will be discussed in Chapter 3, the object-attached feature assumption underpins many of the approaches to recognition and attitude estimation reported in the literature. The recognition framework in this thesis is specifially designed to not invoke the object-attached feature assumption. does not use object-attached features. This is justified by

the severe limitations on the generality of the types of object that can be recognized that result when object-attached features are used. Table 1.1 shows that 3-d recognition methods that rely on object-attached features are restricted to objects that fall into one or more of the following three classes:

- objects with many linear creases;
- objects with creases possessing many points with zero or infinite curvature in the direction of the crease;
- objects with many pointy protuberances where both principle curvatures are approximately infinite.

Many 3-d recognition methods in the literature have concentrated on polyhedral objects (see Chapter 3 for examples), a subgroup of the first class of objects. Similarly, a subgroup of the second class, planar objects, have received considerable attention as well. The third class of objects has received less attention, perhaps because such objects are more uncommon than those in the previous two classes.

In summary, relying solely on object-attached features in a recognition system undermines its chances of achieving two important objectives: robust performance and a large and general object vocabulary. The difficulty in achieving robust performance in a system that employs object-attached features is a consequence of the fact that many robust features, such as those derived from occluding contours, generally cannot be considered object-attached.

The limitations on the vocabulary of a system that employs object-attached features is a consequence of the small and specialized nature of the 3-d surface geometries that reliably project to object-attached features. For these reasons, the recognition framework described in this thesis has been specifically designed so that both object-attached and non-object-attached features may be used. As will be discussed more fully in the coming chapters, this choice complicates the design of the system. However, the additional complexity is deemed worthwhile considering the resulting gains in flexibility and robustness.

1.4.2 The Nature and Complexity of the Object Recognition Problem

Previously, object recognition was defined as a search for model instances that predict well what is observed in the image. As we will show shortly, the space of possible model instances is very large. However, the space that must be searched by the object recognition algorithm is still larger. A model instance describes only geometry, i.e., the model's pose in the scene. While geometry is the primary factor in determining the *shape* of the model in the image, a number of other factors can drastically influence the appearance of the object in the scene. These factors include the spectral and spatial distribution of the illumination as well as any unknown camera parameters (such as the focal length of an adjustable zoom lens).

The complexity of object recognition arises from the huge space that must be searched. If the number of objects that may appear in a scene simultaneously is unbounded, then the size of the space is also unbounded. However, other limitations put a practical limit on the number of objects that can be present, and recognizable, in a scene. For the purpose of estimating the complexity, let o be the maximum number of recognizable object that may appear in a scene. Then, the space of *scene instances* $\mathcal{S}$ that must be searched by a recognition system can be described by a set product: $\mathcal{S} = \mathcal{M}^o \times \mathcal{I} \times \mathcal{C}$, where $\mathcal{M}$ is the set of *model instances*, $\mathcal{C}$ is the set of *camera instances*, and $\mathcal{I}$ is the set of *illumination instances*. As defined previously, a model instance completely specifies the position and orientation of a model in the scene. Analogously, an illumination instance completely specifies the light flux incident on a model in the scene. Similarly, a camera instance completely specifies the behavior of the camera.

Each scene instance $s \in \mathcal{S}$ contains all of the information necessary for a graphics program to produce an accurate rendering the model's appearance in the scene. Let $R(s)$ be an image generated by a rendering process, and I be an image taken by a camera. Also let $D(s) = D(R(s), I)$ be some measure of the *disparity* between the rendered model and the image. The task of any object recognition algorithm is to search through

$\mathcal{S}$ for particular scene instances s_r such that the disparity $D(s_r)$ is sufficiently small. Thus, the recognition task is, in reality, a optimization problem over the set of scene instances $t \in \mathcal{S}$ with $D(\cdot)$ as the objective function.

Each model instance, $m_j \in \mathcal{M}$, $j = 1 \ldots o$ comprising a scene instance $s \in \mathcal{S}$ is *coupled*, meaning that changing the parameters of one model instances, say m_2, could potentially result in changes in the appearance of all other model instances comprising s when s is rendered. The mechanisms responsible for this coupling are occlusion and shadows. While many recognition methods make allowances for handling partial visibility and occlusion, the effects of shadows are usually ignored. In most cases, occlusion is handled as though it is the result of some kind *noise* that causes portions of the object to be randomly missing. This simplified treatment, in conjunction with the suppression of shadow effects allow the model instances to be treated as though they are uncoupled. This simplifies $\mathcal{S}$ to $\mathcal{M} \times \mathcal{I} \times \mathcal{C}$, a potentially much smaller set than the previous one. Fortunately, for many recognition domains, this is a reasonable assumption. Most 3-d recognition systems employ this assumption Cyclops also employs this assumption.

A notion of the complexity of object recognition can be obtained by examining complexity of searching $\mathcal{M}$, which places a lower bound on the complexity of searching $\mathcal{S}$. As described previously, a particular model instance m is an ordered pair (M, T_{wo}) where M is a model identifier and T_{wo} is a transform from object coordinates to world coordinates. Since, for the purposes of this thesis, M is assumed to be rigid, the transform T_{wo} contains six degrees of freedom. A simple parameterization of this transform consists of the six-dimensional vector $(x, y, z, \alpha, \beta, \gamma)$. The translation parameters x, y, and z specify the coordinates in the world frame of some reference point in the model frame, such as, for example, the center of mass. The orientation parameters α and β specify the direction of an axis passing through the reference point and fixed in the model frame. They form a polar coordinate system with α as the "latitude" and β as the "longitude". Therefore, measuring in radians, $\alpha \in [0, \pi]$ and $\beta \in [0, 2\pi]$. The angle γ specifies the

rotation of the model about the axis, hence $\gamma \in [0, 2\pi]$. Also, assume that the volume in space that objects may appear is contained in the cube $x, y, z \in [-1, 1]$. Let there be N_m models. In addition, assume that the application requires that the translation parameters of the model be estimated to within ϵ_t units and that the orientation parameters be estimated to within $\epsilon_o \pi$ radians. Thus, the pose of each model must be resolved within a 6-d hypercube of volume $\epsilon_t^3 \epsilon_o^3 \pi^3$. The entire parameter space has volume $2^5 \pi^3$. Thus, approximately $\lfloor 32/\epsilon_t^3 \epsilon_o^3 \rfloor$ of the resolution volumes fit within the entire space. Hence, the total number of volumes that must be searched is $N_m \lfloor 32/\epsilon_t^3 \epsilon_o^3 \rfloor$. A typical application may require $N_m \approx 100$, $\epsilon_t \approx 2 \times 10^{-3}$ and $\epsilon_o \approx 2 \times 10^{-3}$. This yields 5×10^{19} resolution volumes. Even if evaluating the disparity function D within the volume were computationally cheap, this many volumes is far too large to exhaustively examine. To worsen matters, D is not usually cheap to evaluate.

In view of the large number of possible evaluations of D, the effective number of evaluations must be a very small fraction of the total number of volumes. This implies that the search strategy must be very efficient. There are a number of way to improve the efficiency of the search through the space of scene instances:

- employing a multilevel representation.
- effective application of the viewpoint consistency constraint mentioned above.
- incrementally evaluating D (as in hypothesize and verify, for example).
- o effective low-level grouping processes.
- o employing object-attached features.

Of these, the ones marked with • are critical to the success of any 3-d object recognition algorithm in the sense that it is unlikely that any effective recognition strategy can operate without them. Of the others: good low-level grouping methods improve efficiency but usually do not improve the robustness of the algorithm, and employing object-attached features usually simplifies and speeds recognition, but, as we have seen, at the cost of

considerable generality. Chapter 3 examines how these avenues have been followed in the in the literature.

CHAPTER 2

THE CYCLOPS OBJECT RECOGNITION FRAMEWORK

This chapter provides an overview of a framework for 3-d object recognition. Since this framework is designed to recognize 3-d objects from a single intensity image, we call it Cyclopsafter the mythical one-eyed creature of Greek lore. Certain aspects of the overall framework are investigated in more detail later in this thesis. The purpose of this Chapter is to unify into a coherent whole these aspects that otherwise might not seem so closely related.

2.1 Goals of Cyclops

Specifications for any object recognition system fall into three categories: those concerning the sensor, those concerning the scene, and those concerning the knowledge that the system will possess.

2.1.1 Sensor

- **Sensor Type:** Video camera (incident light flux sensor).
- **Camera Resolution:** The camera has resolution greater than 256×256 in the field of view.
- **Camera Noise:** The camera may be noisy.

2.1.2 Scene

The scene related goals may be further decomposed into those concerning the nature of the objects in Cyclops's vocabulary, those concerning constraints on the geometry of the placement of objects in the scene, and those concerning lighting in the scene.

Vocabulary

- **Shape:** Objects may have any shape.
- **Rigidity:** Objects are assumed to be rigid, with clear extensions to more general types of models.
- **Surface Reflectance:** Markings and specularity should be tolerated as well as possible.

Scene Geometry

- **Pose:** General pose allowed.
- **Number of Objects in Scene:** Large number of objects allowed.
- **Unknown Objects:** Objects that are not in the vocabulary may appear (though the Cyclops will not recognize them).
- **Occlusion:** Objects may be only partially visible due to occlusion by other objects, distortion by noise, and other phenomena.
- **Background:** Objects may appear in the midst of highly cluttered background.

It is worthwhile to note that the final item above in the scene geometry specifications makes it impossible to assume figure-ground segmentation *a priori*, as is done in some algorithms (see Chapter 3).

Scene Lighting

- **Uniformity:** Lighting may be non-uniform, possibly from multiple sources.
- **Shadows:** Must tolerate shadows as well as possible.

2.1.3 Knowledge

- **Camera Model:** Camera modeled as a pinhole camera under weak perspective, with clear path for extension to full perspective.
- **Object Models:** Object models are detailed geometric models, with mechanisms for extension to models with surface reflectance information.

2.2 Overview of Cyclops

Fig. 2.1 shows the breakdown of computation in the Cyclops framework. The computation is divided into major processes. The data structures that these processes operate upon are shown in the figure as well. These processes are broken down below.

- Image Representation Processing:
  - **Feature detection.**
  - **Feature grouping.**
- Matching:
  - **Hypothesis Generation.**
  - **Hypothesis Refinement:**
    - ⋆ **Incremental Verification.**
    - ⋆ **Attitude Estimation.**

The data structures are closely linked to the processes that operate on them: they must be chosen to allow the processes accessing them to operate as efficiently as possible. Some of the data structures can be built "offline", i.e., building them does not detract from the system's runtime performance. For such data structures, we have the freedom to trade large amounts of computation offline for potentially large improvements in the efficiency of runtime recognition. This is precisely what we have done in Cyclops, resulting in a great improvement in the efficiency of the entire matching process.

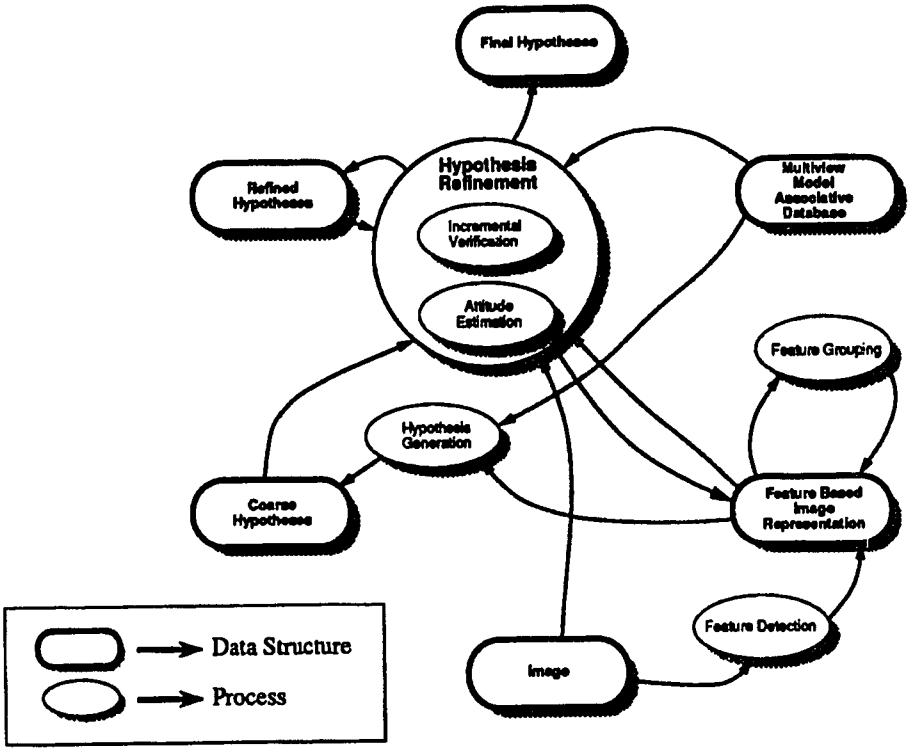


Figure 2.1. Relationship between processes and data structures in Cyclops.

The primary focus of the research leading to this thesis has been on the processes and data structures related to *matching*. Cyclops's approach to matching is what differentiates it from exisiting 3-d object recognition approaches. There is a simple explanation for this: *Cyclops was specifially designed to not use the object-attached feature assumption.* This, in turn, has resulted in the development of new methods for performing matching. While we have done considerable research into matching we have done some additional work on investigating processes and data structures related to *image representation* as well.

The complexity of Cyclops due in large part to the design goal of employing *any* type of features, not just object-attached features, has precluded in-depth research on all the processes that comprise Cyclops within the confines of this thesis. The processes that received the most attention, *hypothesis generation*, *attitude estimation*, and *feature detection*, and their associated data structures, are those that were most affected

by the decision to use general features. The *feature grouping* and *incremental verification* processes recieved less attention, as these processes are largely independent of the object-attached feature assumption. Thus, for example, previous work on grouping remains valid, even in absence of the object-attached feature assumption. The remainder of this chapter describes all of the processes of Cyclops focussing on the aspects that are new and unique.

2.2.1 Features

In order to perform their tasks effectively, various object recognition processes in Cyclops employ a number of distinct representations of the 2-d shape of both image and model edges. Such representations are sometimes called features. The tasks performed by the processes differ, so it is not suprising that the optimal shape representation differs between processes. For example, we discuss later how the hypothesis generation process works best with a *spatially sparse* representation that is very *selective*. In addition, it is helpful if such representations are invariant or insensitive to variations in as many of the viewing parameters as possible. Cyclops employs configurations of *critical points* and *inflection points* and representations of the local shape near these points. Critical points are points of high curvature along an edge contour, whereas inflection points are points of zero curvature. This type of relatively incomplete shape representation corresponds closely to the conventional notion of a feature. By contrast, the processes that constitute the hypothesis refinement portion of Cyclops require a more complete representation.

2.2.2 Models

As is the case with most existing 3-d recognition approches, Cyclops requires 3-d geometric models of the objects in its vocabulary. The question arises: what is the best representation for a 3-d model so that it best facilitates recognition of it in an image?

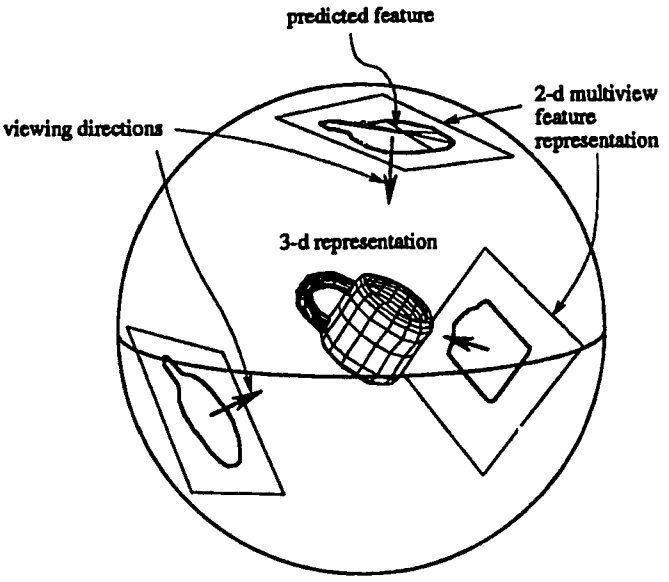


Figure 2.2. The multiview model of a coffee cup. The model consists of a 3-d model and a 2-d multiview feature representation. The 2-d multiview feature representation consists of an image-domain representation of a set of projections of the 3-d model from various viewpoints, such as the three shown. The features from these projections can be indexed in a data structure for fast matching.

We have designed Cyclops to employ *multiview models*. A multiview model consists of two parts, shown in Fig. 2.2:

- 3-d representation.
- 2-d multiview feature representation.

The 3-d representation contains enough information to allow the appearance of the edge contours of the object to be predicted, given a set of viewing parameters. The 2-d multiview feature representation consists the features appearing when the object is viewed from each of a set of viewpoints. We choose these viewpoints to sparsely cover the sphere of views. These features encode local shape.

That Cyclops employs multiview models is a direct consequence of our refusal to invoke the object-attached feature assumption. The reasons for this are discussed in Chapter 4. As we would expect, algorithms that rely on object-attached features generally

do not use multiview models. Rather, since using object-attached features implies a correspondence between 2-d image features and 3-d model features, some type of 3-d representation is all that is necessary. Three-dimensional representations are, by their nature, viewpoint independent.

Multiview models first appeared in algorithms that use *global features*. Global features encode the shape of the region enclosed by the silhouette boundary of the object. Examples of such global features include Fourier descriptors, moments, and Wigner transforms. Since the surface of the object visible to the viewer on the object changes with viewpoint, we can conclude that global features are inherently *not* object-attached. This supports the claim that avoiding object-attached features requires an algorithm to employ some type of multiview model.

In previous algorithms employing multiview models, the models consisted only of a list of pairs (ν, f), where ν is a viewpoint and f the associated global shape feature resulting from viewing the object from viewpoint ν; the 3-d representation is typically either discarded, as in [KD87], or never actually existed, i.e., images of real objects at different viewpoints were processed to obtain a multiview feature-based representation as in [DBM77]. In the following section, we describe how such multiview representations are related to *aspect graphs* which are based on the topology of the features in the image.

In Cyclops, the multiview model has been extend to use local features, a necessity if the system is to recognize objects having parts that are missing or distorted. In addition, the multiview models used in the Cyclops framework are hybrids, consisting of both the 2-d multiview feature representation and the 3-d representation that is used to generate the features from the various viewpoints. This is because, as in the case of features and shape representations, various processes work most effectively with different types of representations. As will be shown later, the efficiency of the hypothesis generation process is greatly enhanced by utilizing a multiview 2-d feature representation. On the other hand, for example, the attitude estimation benefits from the use of a 3-d representation.

2.2.3 Hypothesis Generation and the Model Database

Conceptually, the function of the hypothesis generation process in Cyclops is straightforward: given an image feature, find all model instantiations that could result in a feature that is sufficiently similar to the image feature. Implementing this simple statement *efficiently* is not simple, however.

As indicated above, an important part of hypothesis generation is the process of determining which features are similar and which are not. Intuitively, similar features will have similar attributes, i.e., their feature vectors will have similar components. This implies that if the distance between two n-d feature vectors is small, then the features will be similar in appearance. Thus, if $\mathbf{f}_i$ is an n-d feature vector that has been detected in the image, and, similarly, $\mathbf{f}_m$ is an n-d feature vector that has been predicted from a particular model instance, then $\mathbf{f}_i$ and $\mathbf{f}_m$ are similar if $\mathbf{f}_m$ falls within a *neighborhood* of $\mathbf{f}_i$. We would not expect $\mathbf{f}_i$ be identical to $\mathbf{f}_m$ even if they are correctly matched since noise and other distortions present in the imaging process would alter the attributes of $\mathbf{f}_i$ from the "noiseless" attributes of $\mathbf{f}_m$.

In contrast to object-attached features, predicting non-object-attached features from the 3-d representation of the model is often computationally expensive. This is because predicting general features usually involves computing some kind of *rendering* of the model whereas predicting an object-attached feature usually involves nothing more than and projecting some 3-d feature on the model's surface. Rendering a model is usually a time-consuming undertaking. Therefore, much time can be saved during the recognition process if some of the prediction of features is done offline. As discussed above, these offline predictions of features make up the *multiview feature representation* part of the model. Ideally, such features are computed throughout the space of viewing parameters wherever a significant visual event occurs when traversing a path from one sample to a neighboring sample. Such visual events may include, among others: large changes in the attributes of any of the features, and appearance or disappearance of features.

This is a generalization of the concept of *aspect graphs* [KvD79, SB87]. Aspect graphs typically partition a set of views of an object by the *topology* of the features (usually the projections of tangent and jump discontinuities on the object's surface, structures that tend to generate edges when imaged). Topological representations are insensitive to changes in viewpoint, resulting in a relatively small number of "aspects". This is an advantageous property for a representation to possess. Were it necessary to sample the entire viewing parameter space, the number of model instances that would have to be represented would be intractable. Fortunately, if invariant features are used, only a small subspace of the entire viewing parameter space need be sampled. Topological representations are not suitable because errors in segmentation often result in gross changes in topology.

The features employed by Cyclops's hypothesis generation process have been designed to be invariant or insensitive to as many viewing parameters as possible: they are invariant to image-plane rotation, and translation; they are invariant to image plane scaling over wide ranges of scale; and, finally, they are insensitive to lighting variations. These features, described in detail in Chapter 4 consist of configurations of shape descriptors of edge contours in neighborhoods of high-curvature points (critical points) and zero-curvature points (inflection points). Fig. 2.3 shows an example of a feature formed by a critical point and an inflection point. The attributes of these features consist of invariant shape descriptors, pose descriptors, and scale descriptors. For example, an invariant shape descripor would be the angle between the line segment from the center, c, to a_0 and any one of the line segments $ca_{\pm 1}$. Pose descriptors include the orientation of the line segment a_0b_0. A scale descriptor is the length of a_0b_0.

The invariant properties of these features imply that the features constituting the multiview feature representation part of the multiview model need be predicted only over the surface of the viewing sphere, as opposed to throughout the entire multidimensional viewing parameter space, as illustrated in Fig. 2.2.

One of the reasons for choosing to use multiview models is that the multiview feature

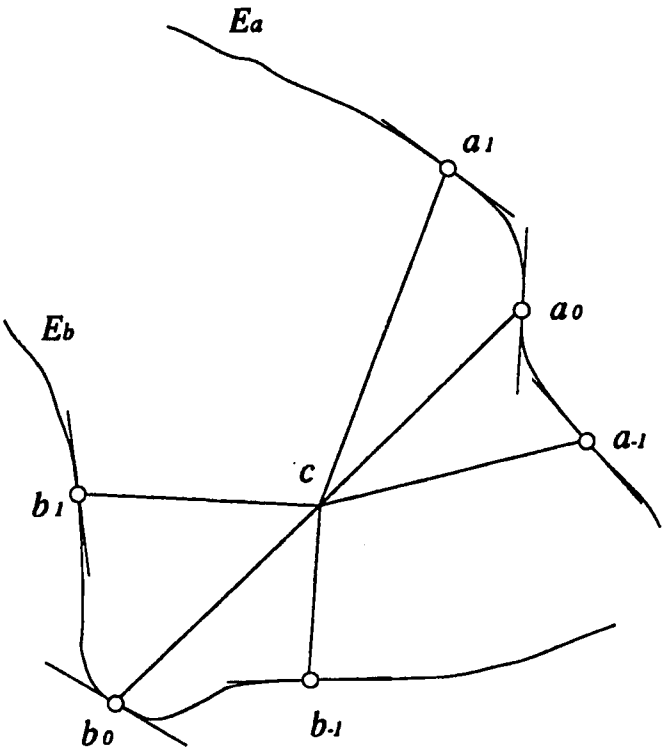


Figure 2.3. An example of a feature used in the Cyclops framework. in this case a pair of primary points consisting of high-curvature point, a_0, and an inflection point, b_0, and four auxliary points, $a_{\pm 1}$ and $b_{\pm 1}$, derived from the primary points, and the tangents to the edge contours at all of the points.

representation allows the predicted features visible from various viewpoints to be stored in a *model database*. This database is indexed so that the predicted features that are *similar* to an image feature, i.e., predicted model features falling in a neighborhood of the image feature in feature-attribute space can be retrieved in minimum time. In Chapter 4, we show how to perform this operation with $O(\log(N))$ complexity, where N is the number of features stored in the database. Study of recognition algorithms that have used precomputed predicted features to generate hypothesis reveals, suprisingly, that no other algorithms have taken full advantage of the freedom to index these features. In fact, with the exception of some sub-linear examples, the typical algorithm is linear in N (none better than $O(N^{\frac{1}{2}})$). Considering that N may easily be larger than 10^4 for a moderately

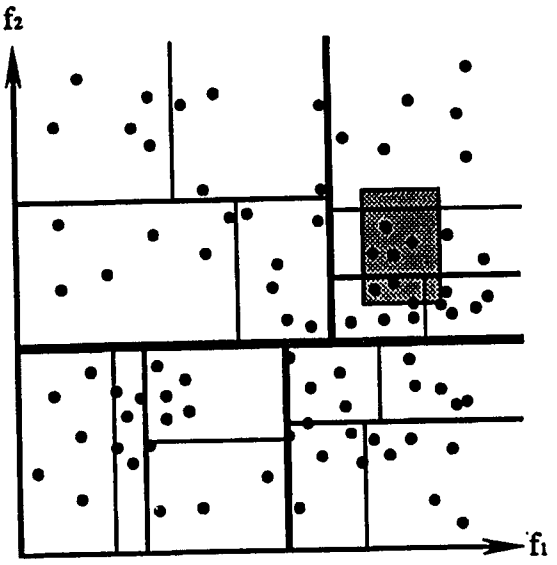


Figure 2.4. A hypothetical 2-d feature space partitioned by a k-d tree. Each dot represents a feature vector. Each line represents a node in the tree. Thicker lines represent nodes closer to the root. Queries represent arbitrary rectangles. A query is processed by traversing the partition boundaries from thicker to thinner until the cells containing the query rectangle are reached. This can be done in average case $O(\log N)$ time, where N is the number of features.

sized model vocabulary, and considering that hypotheses may be generated for a sizable fraction of the features present in the image, achieving the highest possible performance from the hypothesis generation process is critical to the success of the algorithm.

Cyclops's model database is implemented with a data structure called a *k-d tree*. A k-d tree hierarchically subdivides a k-dimensional feature-attribute space with half-planes until a small number of feature vectors occupy each leaf cell. This is shown for a hypothetical 2-d feature space in Fig. 2.4. The $O(\log N)$ query time results from the fact that only $O(\log N)$ divisions are needed to reach a given cell.

In essence, the model database based on a k-d tree is an instance of a general vector associative memory, which is also extremely efficient. Thus, the utility of a structure like the model database extends to many other pattern matching tasks in computer vision and pattern analysis where efficient associative matching is useful.

2.2.4 Viewing Parameter Estimation

The user of the results of the recognition process typically requires the pose of the recognized objects as well as their identity. In addition, a key part of model-based tracking consists of pose estimation. These are two strong motivations for developing robost viewing parameter estimation techniques. However, are others as well. Since the model database, described in the previous section, is a finite sampling of the viewing sphere, the error in the viewing parameter estimate contained in a newly retrieved model instance could be as large as the distance between it and its nearest neighbor on the viewing sphere. This implies that the shape of a hypothesized model instance may differ considerably (depending on the density of samples on the viewing sphere) from the actual model instance appearing in the image, assuming that the model identity is the same. Due to the potential of a considerable shape disparity induced by differing viewing parameters, it is impossible to do a detailed verification of the hypothesized model instance. Therefore, it is critical that the important viewing parameters can be estimated so that differet hypotheses may be considered on an equal footing.

Previous approaches to the attitude estimation problem have a number of shortcomings. Many approaches use the object-attached feature assumption. In other words, such approaches form enought correspondences between these two sets of features, that, *so long as the correspondences are correct*, it is possible to solve for the viewing parameters that map the 3-d model features to the 2-d image features. In such cases, the solution can be found analytically in the case of weak perspective, as in [Hut88], by iterative methods, as in [Low87a], or by successive refinement [Goa83]. Aside from the problematic use of object-attached features, assuming a correspondence implies that a search must be conducted through the exponentially large space of correspondences between 3-d model feature and 2-d image features, perhaps with a time-consuming fit at each step. Other approaches employ optimization over the viewing parameters to minimize measure of disparity between global shape descriptors, such as Fourier descriptors or moments,

obtained from the model and the image. The use of global features is severely restrictive, since it implies that the object has already been segmented from the background. If it is possible at all, performing such segmentation bottom-up is probably more difficult than recognition itself. In addition, global features preclude the ability to recognize objects that may be only partially visible.

The approach to attitude estimation in Cyclops, which we call *Attitude Estimation by Feature Modulated Attractors*, or AEFMA, addresses all of the problems mentioned in the paragraph above:

- it works with any 3-d object, not only those that generate large numbers of 2-d object-attached features,
- it does not make correspondence hypotheses between predicted and detected features,
- it can estimate viewing parameters of partially visible objects, and
- it does not assume figure-ground segmentation.

The method is based on optimization of a carefully constructed image-to-model shape disparity functions. AEFMA is based on the intuitive notion that, given a set of features predicted from a particular model instance, then the viewing parameters should be adjusted in order to reduce the disparity between the model features and the image features that are most *similar* to the model features. Fig. 2.5 shows a simple way to understand AEFMA. If the disparity function is thought of as a physical pseudo-energy potential function, then the gradient of this function can be thought of as a pseudo-force acting on the model. When viewing parameters are adjusted so that the model is in equilibrium, i.e., there is no net force on it, then a minima of the disparity has also been attained. The net pseudo-force on the model is composed of the pseudo-forces acting between each 2-d predicted feature and each 2-d image feature. The key idea is that the attractive force between a predicted feature and an image feature is proportional to their similarities, i.e., the force is *modulated* by the similarity in the feature attributes.

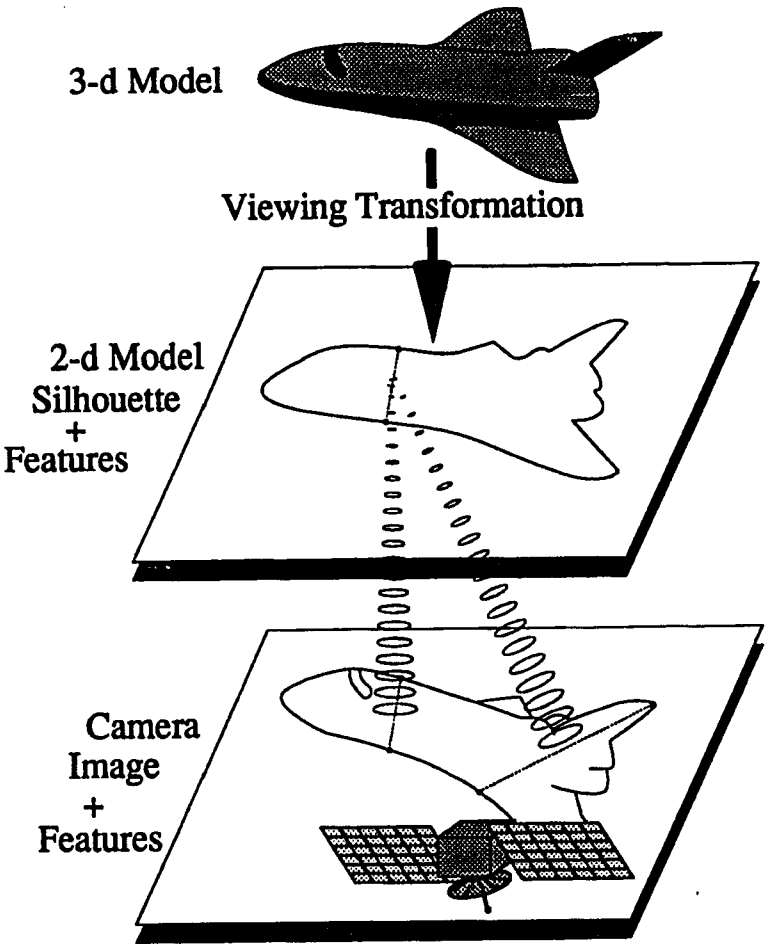


Figure 2.5. For a particular viewing transformation, the 3-d model is used to predict which edge contours will be present when the object is viewed (prediction plane). Similarly, the edge contours in the image are detected. The shape of the edge contours, both predicted and detected, are represented by a set of shape primatives. For each model primative in the prediction plane, such as the one shown, a disparity measure, is calculated for each image primative in the image plane, such as the two shown. All such pairs are summed to form a composite disparity function which can be used to optimize the viewing parameters to obtain the viewing parameters resulting in the best match between the shape of the predicted edge contours and the shape of the image edge contours.

The idea that the strength of the attractive force between a predicted feature and an image feature should be modulated by their similarity has a number of advantageous

consequences. First, the strength of the modulation can be thought of as a "fuzzy" degree of correspondence between image features and predicted features. Thus AEFMA is able to dynamically determine the correspondence between predicted features and image features yet avoid the combinatoric explosion inherent in a discrete search. Second, modulation by feature similarity reduces the likelihood that the potential function will contain misleading local minima that could trap the optimization procedure. The reasons for this are beyond the scope of this chapter. Chapter 5 contains details. However, the importance of reducing the number of local minima in the potential function cannot be overemphasized.

The form of the potential function is critical to the success of the algorithm. The pseudo-energy potential is obtained by summing the contributions of the potentials of each pair of features, one of which is derived from the projection of the model from the current viewpoint, while the other is derived from the image. Each of these terms is an Interfeature Disparity Function, or IDF. The form of the IDF is very important. For example, unlike the $\frac{1}{r}$ potential law generated by such physical forces as the gravitation and electromagnetism, the pseudo-force potential cannot possess a singularity, as is the case with the 2-d potential shown in Fig. 2.6. This is because the model has no "mass", and therefore, if a predicted feature approaches too close to an image feature, the model feature will become trapped, even if the match is not good. In Chapter 5 we show that the potential function should have a smooth, flat center. Additionally, we show that if the feature attributes are widely different, the force should be small, and therefore the potential function should be near horizontal. These extremes should be smoothly blended together so that the force increases up to a maximum distance is reached and then decays to zero, deemphasizing the importance of features that are too distant in feature-attribute space. A potential function that possesses these properties is multidimensional Gaussian distribution. An example of a symmetrical 2-d Gaussian is shown in Fig. 2.7.

A particularly important property of the multidimensional Gaussian is that the minima

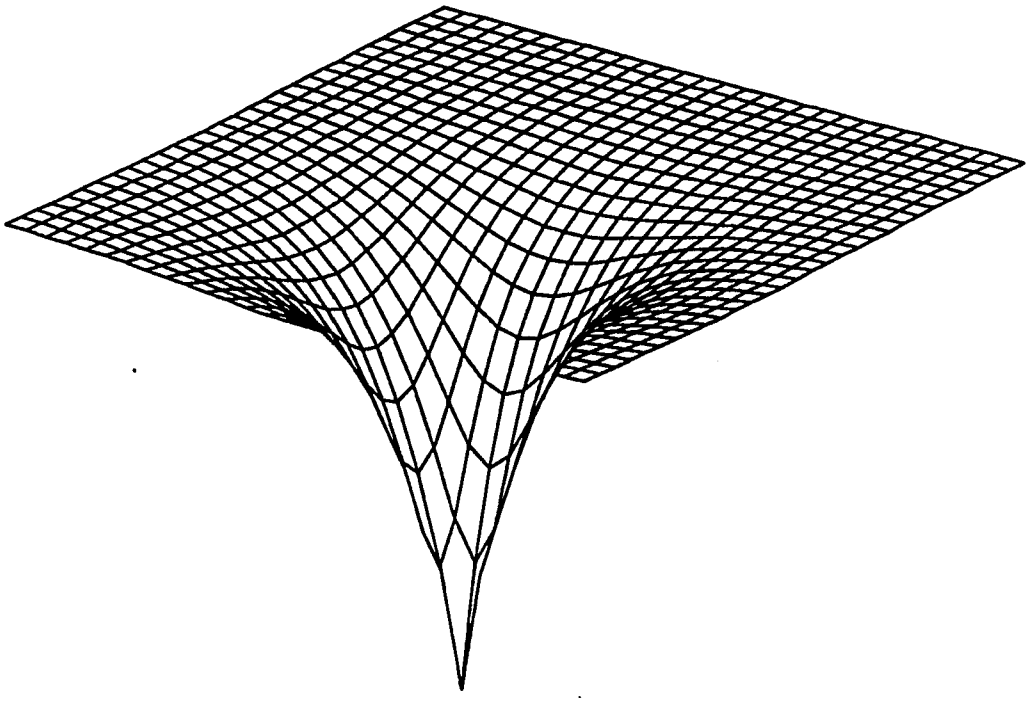


Figure 2.6. An example of a potential function that is unsuitable for the IDF because is has a singularity at its center.

of a sum of such Gaussians behaves in a predictable manner. In particular, the sum of multidimensional Gaussians tend to blend together into a function possessing a single minima, even when the Gaussians are relatively far apart in the feature-attribute space. This is important because it implies that spurious local minima will tend to be suppressed in the summation. As we will show in Chapter 5, this is not true of many possible potential functions, such as the one in as shown in Fig. 2.6.

By varying the sigma parameters of the multidimensional Gaussian potential functions the accuracy of the estimate of the viewing parameters can be traded for wider range of convergence to the viewing parameters: if the sigmas are large, the minima of the composite potential function will be flat and not well localized. However, there are few local minima near the global minima (within one sigma or so). Thus it is easy to locate the global minima of the composite disparity function. Unfortunately, the use of a large sigma causes the minima of the composite disparity function to be perturbed by

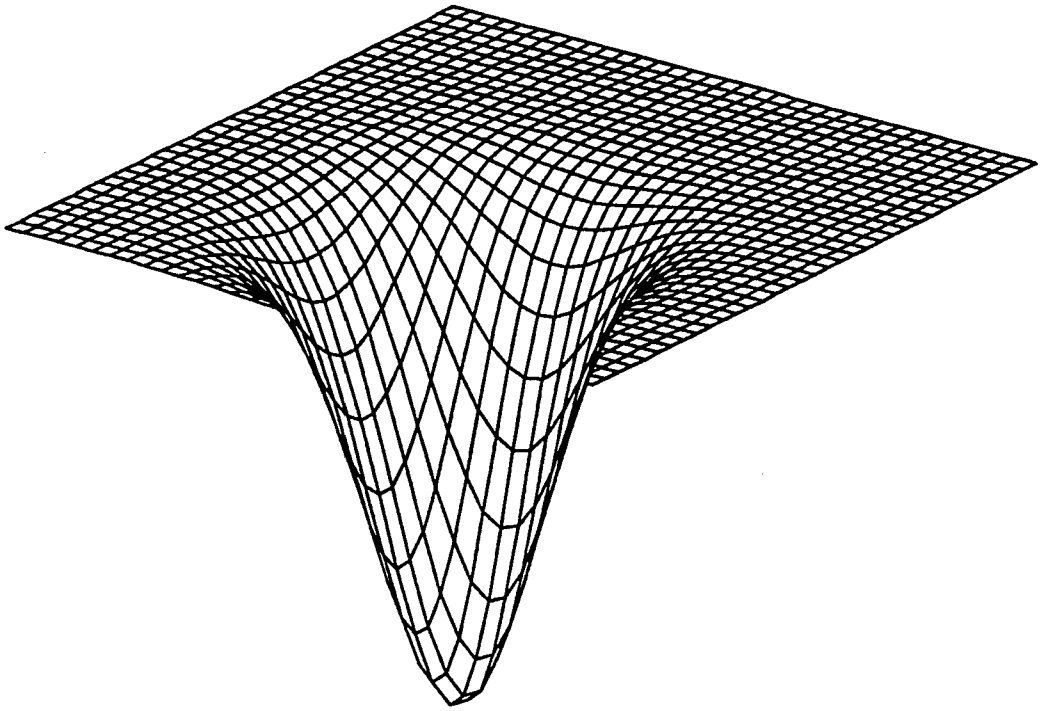


Figure 2.7. A 2-d Gaussian, the type of function that Cyclops uses as the IDF.

the effects of distant features that may not belong to the correct instance of the model in the image. If sigma is reduced, the localization of the minima is improved. However, the improvement comes at the expense of reducing the size of the volume of viewing parameter space in which the optimization procedure can be expected to converge to the global minimum. Thus, to achieve the good localization of a small sigma with the large range of convergence and few local minima of a large sigma, the method is iterated, starting with a large sigma, and the successively reducing sigma and re-solving until the desired viewing parameter accuracy is achieved.

Fig. 2.8 shows the result of running an implementation of AEFMA on an image of the space shuttle.

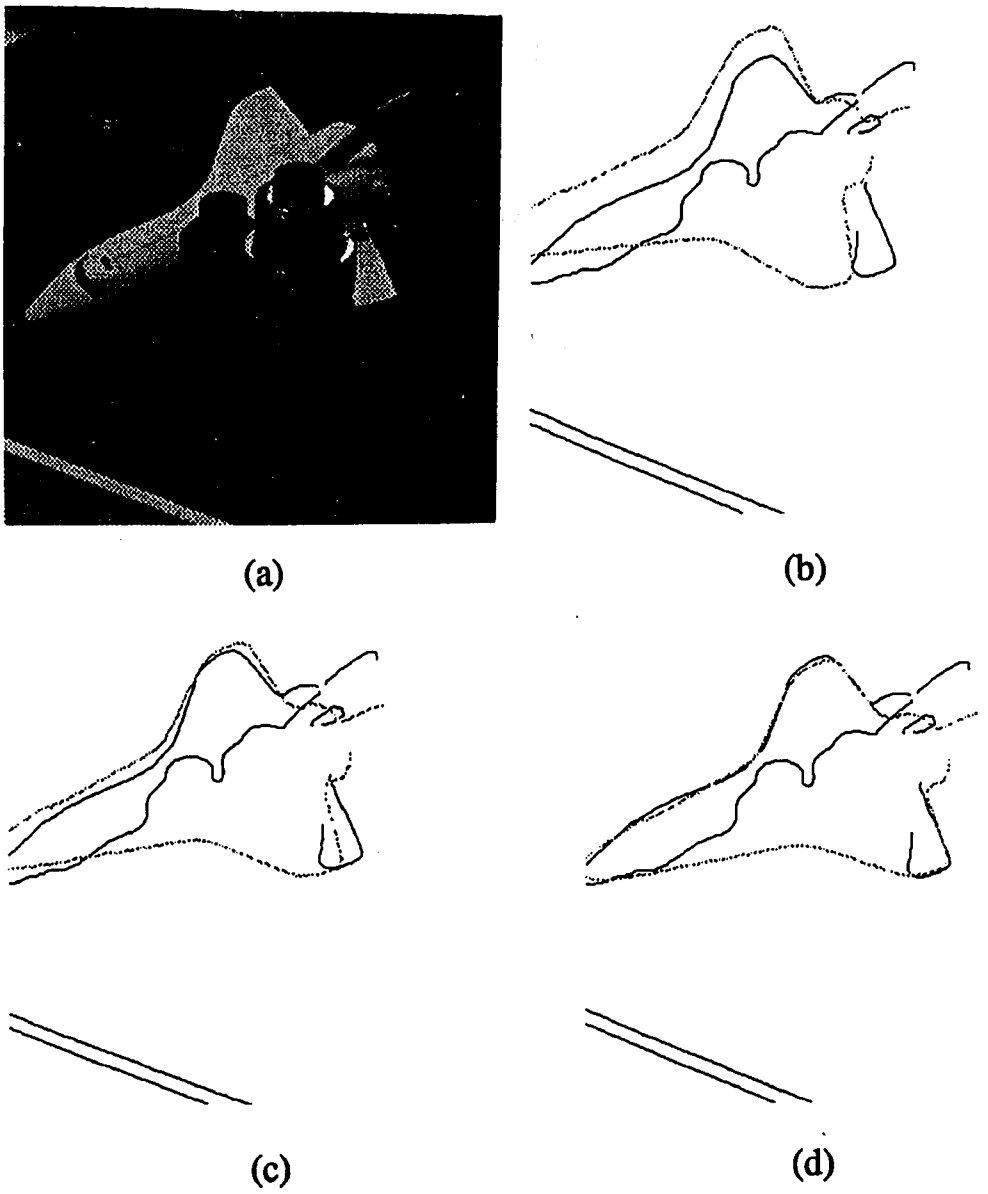


Figure 2.8. The result of running AEFMA on a real image of the space shuttle, (a). In (b) is the result of edge detection on the image (black contours) along with the contours predicted from a model in its initial pose (dotted contours). An intermediate result is shown in (c), and the final result is shown in (d).

2.2.5 Incremental Verification

The idea behind incremental verification is to reduce the overall effort expended on

verifying hypotheses by performing the verification in stages, reserving the greatest effort for the most promising hypotheses while weeding out the less promising hypotheses early. In this respect, incremental verification bears some relationship to techniques which use tree structured searches, e.g. [Bro81, GLP85, AF86], which successively match model feature to image features, pruning the paths that have prove to be inconsistent with the model, until there is either a single consistent interpretation, or there are no consistent interpretations remaining. In such algorithms, each potential model-feature to image-feature match that proves to be consistent is positive evidence for the interpretation.

In classical "hypothesize and verify" algorithms, e.g. [BC82, KJ86, GTM87, CA87], the verification is completed in one step, after which a decision as to the fitness of the match is made. Most verification algorithms in the literature use a very *complete* representation of the image and the model instance. This is simply because a more complete representation allows more predictions to be made and tested. Such predictions with supporting observations are precisely the evidence that the verification uses to make decisions about which hypotheses to pass.

Computationally, verification is usually a fairly expensive undertaking. This is because verification algorithms tend to operate near the image in terms of data abstraction, and the volume of data that must be processed is correspondingly large. In contrast, the tree-structured search methods mentioned in the first paragraph typically operate at the feature level, and often require less computation per hypothesis than hypothesize and verify algorithms.

Are there clear advantages to either of the paradigms above? A well-implemented tree search method is often faster, but it is often not as robust at the hypothesize and verify paradigm. Verification can often be accelerated with simple hardware, such as a pipelined processor, since the operations are usually simple and near to the pixel level whereas the tree search method is more suited to slower, general purpose processors. On the other hand, it is wasteful to expend a large amount of effort to verify and reject a poor

hypothesis when a small amount of effort would reveal the most promising candidates. In general, the tradeoffs are so complicated and algorithm dependent that no effort is made here to find the "optimal" approach. More likely, there are many possible "very good" approaches. The incremental verification method is designed to combine the incremental, speedy nature of the tree search paradigm with the robustness of the hypothesize and verify methods.

Unlike the tree structured search approaches, which typically operate in a homogeneous space of some type of features, such as 2-d line segments and 3-d planar facets [Goa83], or trapezoids and ellipses [Bro83], incremental verification takes heed of the following observation: the more model information that is used during verification, the more robust the decisions to accept or reject a model will be. Using a relatively incomplete representation of the object by sparse features does not yield robust verification. The essence of the incremental verification method is to have several *levels* of verification. Cyclops is designed to have the following degrees of verification

1. Newly generated hypotheses.
2. Hypotheses after gathering evidence of any loosely matching features.
3. Hypotheses after gathering evidence of features matching tighter criterion following 2-d refit.
4. Hypotheses that have been refined by AEFMA, using final objective function value.
5. Passed after final point-by-point boundary verification.

2.2.6 Grouping

Grouping mechanisms occur at many points in object recognition algorithms, from the lowest grouping of pixels into regions or edges to grouping instantiated sub-objects into whole objects. Such processes can be divided into *model-driven grouping* and *model-independent grouping*, where "model" refers to object models. That is, model-independent

grouping consists of grouping that is done solely on the basis of information that is not specific to particular object models whereas model-driven grouping uses relations that are specific to a particular object. Model independent grouping is sometimes referred to as *segmentation*, although segmentation is really a broader term since it can be model driven. In Cyclops, grouping is concerned primarily with determining if the pairs or triples of primitive features (critical points, inflection points, and line segments) that make up a compound feature are likely to have been generated by the same object. For example, "Two features that are joined by a continuous portion of edge contour are more likely to have come from the same object than if they are not," is an example of a model-independent grouping heuristic.

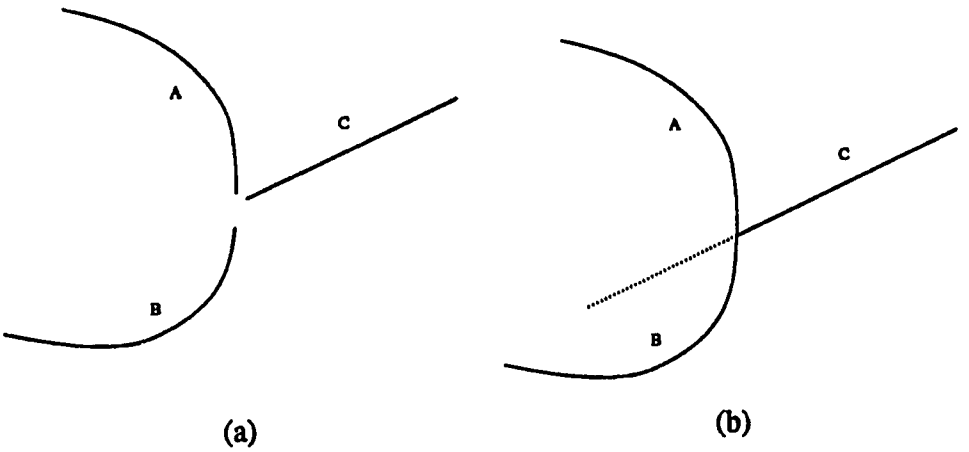
If model-independent grouping information is available, considerable gains in Cyclops's efficiency can be realized. In the absence of any grouping information, then all possible pairs and triples of primitive features in the image data must be considered on equal footing. For example, if there are N primitive features, and if the compound features consist of pairs of primitives, then N^2 compound features must be considered by the hypothesis generation process. On the other hand, if perfect grouping information is available, i.e., it is known *a priori* which features belong to the same object, then many fewer configurations need be considered for possible hypothesis generation since most of the possible pairs could be rejected as invalid. Such invalid pairs have one or more of their features are from distinct objects, from background, from non-vocabulary objects, or are spurious noise-generated features. Hence, from the standpoint of efficiency, grouping information can be very helpful.

While utilizing grouping information can improve efficiency, it must be used with caution: it is often unreliable and error-prone. For this reason, Cyclops uses the grouping information only to prioritize which compound features should be computed, and, further, which compound features should be used by the hypothesis generation process to generate new hypotheses. Thus, good grouping information will speed recognition, while

poor information may slow recognition, but will not prevent an object to be recognized correctly. Contrast this to the numerous systems that rely heavily on model-independent grouping, [BC82, Low87a] for example. When the grouping mechanism fails in such systems, then recognition will often fail as well.

Model-independent grouping of features is most often done by exploiting *spatial proximity* or *spatial continuity* of some image property. For example, features on the same edge contour have favorable evidence for their belonging to the same object by virtue of the continuity of the contour. Unfortunately, edge detectors often break contours at points of high curvature and at points where multiple contours intersect, robbing the recognition algorithm of much potentially useful grouping information.

Overcoming breaks in edge contours is one of the primary purpose of Cyclops's current grouping module, which is described in Section 4.3.2.7 and in Appendix A. The first stage of the grouping assigns connection values between contours using heuristics that are based on the local geometry of the contours near junctions. For example, refer to Fig. 2.9(a) showing three contours whose endpoints are near to each other. From the figure, it appears that two of the contours should be joined to form a smooth curve and the third contour has been generated by a surface that has been occluded by the surface that generated the first two contours, as shown in Fig. 2.9(b). Since the extrapolations of the two contours that should be joined would fit each other well, a high degree of connection would be assigned between these two contours. The third contour would have a small degree of connection between itself and the first two contours because it is likely to have been generated by a surface on a different object. The second stage of the grouping propagates the connection values globally through the image based on the connections at each locally determined potential junction. Contours that are highly connected are considered more likely to have arisen from the imaging of a single object, and, therefore, features generated from them are given higher priority than other features.



(a) (b)

Figure 2.9. A typical situation after edge detection is shown in (a). Contours A and B are derived from the occluding boundary of a single surface that also occludes the surface generating contour C (shown extending behind the surface generating contours A and B by a dotted line in (b)). The correct grouping is to consider A and B as being highly likely to have been generated by surfaces on the same object, while C is much less likely to have been generated from the same object.

2.3 Major Contributions

- Object-attached features are revealed to severely restrict the types of objects that an algorithm can recognize.
- A recognition framework that does not invoke the object-attached feature assumption is designed.
- The problem of viewing parameter estimation using local, non-object-attached features is solved.
- Multiview models of objects are shown to be essential for recognition of objects without the use of object-attached features.
- A logarithmic complexity technique for associatively matching image features to predicted features from multiview models is developed.

- The subtle interplay of feature selection and matching is analyzed.

CHAPTER 3

PREVIOUS APPROACHES TO OBJECT RECOGNITION

Work on 3-d object recognition was launched by the seminal work of Roberts [Rob64]. Earlier work on object recognition dealt primarily with the recognition of 2-d patterns in images. Unfortunately, 2-d object recognition problem is fundamentally different from that of 3-d object recognition, consequently many approaches that work well for 2-d cannot be extended to 3-d. Roberts' work addressed many of the key issues involved with 3-d recognition and, for a first attempt, was remarkably complete. It will be enlightening to examine Roberts' work in some detail later in this chapter. For now, we note that his method uses a combination of viewpoint-invariant qualitative topological relations between features and quantitative, viewpoint-dependent interfeature relations to select possible models and determine their spatial pose relative to the viewer. It was recognized that one of the weaknesses of Roberts' algorithm was the low-level feature extraction and grouping modules. Reasoning that the state-of-the-art in the low-level aspects of vision would eventually catch up, studies of the higher-level aspects of the recognition process were undertaken in simplified artificial domains. Such domains, often referred to as *blocks world domains* [Guz69, Huf71, Wal72, Tur74, Kan78] are usually restrictive about the types of objects that are allowed. A common example is polyhedra.

In synthetic, blocks world-like domains, features and their relationships are assumed to be known precisely, circumventing the difficulties that Roberts encountered with his low-level modules. Thus freed from the difficulties of noisy and error-prone low-level data, researchers in blocks world-like domains found that topological constraints, which

are more viewpoint invariant than most other image relationships, can be used to perform many image analysis tasks, including object recognition. Unfortunately, the state-of-the-art in the low level aspects of vision has never achieved the low error rates that would allow these methods to be used with real data.

While part of Robert's work was carried to unfruitful ends, the rest of Robert's work contains many of the essential components of a contemporary 3-d object recognition algorithm. It is interesting to speculate what the current state-of-the-art might be had the lead provided by Robert's been followed more fully.

3.1 Classification of Object Recognition Methods

As there are so many approaches to recognition, it is difficult to find any single taxonomy that fits all of them well. Perhaps the broadest distinction between methods is based on the relationship beween the sensed data and the object models. Later in this section, how this relationship influences the design of an object recognition algorithm will be discussed.

Recognition systems may also be characterized by the nature of their commonly held attributes. In particular, in this section, systems will be compared based on the nature of their *features*, their *object models*, and, most importantly, their *matching methods*.

3.1.1 The Relationship Between Object Models and Sensed Data

Object recognition algorithms are most obviously divided into two general categories based on the relationship of the sensed data to the models in the recognition system's vocabulary. One class consists of *matched dimension domain* (MDD) algorithms [1]. Such algorithms assume that the scene geometry can be sensed so that geometrical relations

---

1 This terminology was introduced in [Hut88].

in the scene are isomorphic to geometric relations in the models. MDD algorithms include methods for recognizing 2-d objects from intensity images as well as methods for reconizing 3-d objects from range images. In both cases, the geometry of the sensed scene can be directly compared to the geometry of the models in the system's vocabulary. The other class of methods, which we will call *general domain* (GD) methods, consist of systems that assume that the sensors do *not* give explicit geometric information about the geometry of the scene. This class includes intensity-based 3-d object recognition, the topic of this thesis.

Most object recognition methods in the literature are of the MDD variety. This is probably due to the fact that solving MDD recognition is simpler than solving GD recognition. In a matched dimension domain, relative geometrical relationships existing between parts of the models are preserved in the sensed data, to within noise and visibility constraints, regardless of the viewpoint and pose of the object in the scene. By contrast, accomplishing recognition in general domains is more difficult since the geometry of the scene is not directly available from the image, and must be deduced indirectly.

MDD methods fall into two categories: those that are extensible to general domain recognition and those that are not. Our focus is on the solution of the object recognition from intensity images, which is a case of a general domain problem. Thus, we will concentrate on those methods that contribute to understanding or solving such problems. We will briefly examine the techniques that are not extensible to the general domain, primarily to gain understanding of what makes them inextensible.

3.1.2 Anatomy of Machine Recognition

Object recognition algorithms may be classified according to the particular nature of their commonly held attributes. In particular, recognition algorithms can be compared by the nature of their:

- **models:** how objects are represented to facilitate recognition.

- **features:** how the sensor data is transformed and grouped into representations that facilitate recognition.
- **matching:** how the space of scene instances is searched for explanations of the sensor data.

We will be most concerned with comparing recognition algorithms on the basis of matching, since this is the focus this thesis. However, the features and models that algorithms use often strongly influences the matching strategy. Therefore, issues associated with modeling and feature selection relevent to matching will be discussed where appropriate.

3.2 Previous Work in Object Recognition

Much of the remainder of this chapter examines and classifies previous work on object recognition. In many cases, classification is not obvious since many object recognition methods combine elements of multiple matching strategies. In addition, there are far too many algorithms in the literature to discuss each of them in full detail here. In order to do a broad survey yet benefit from the insights resulting from detailed inspection of previous work, we will: first define several archetypical matching methods, then discuss one or two examples of each archetype in detail, and, lastly, briefly discuss other, similar, methods. Approaches that have elements of more than one archetype will be discussed where they fit best.

GD object recognition algorithms (i.e., not MDD algorithms) must search the space of scene instances. MDD recognition methods, on the other hand, exploiting the isomorphism between the sensor data and the models, need not search this space. Instead, they may search the space of image-feature to model-feature correspondences, looking for consistent sets of features among the model-features and the image-features. GD recognition approaches usually employ one of the following six archetypical matching paradigms:

- transformation clustering
- hypothesize and verify
- predict-observe-backproject
- backprojection
- global feature-based
- optimization-based

MDD approaches are a more varied lot. Many of them fall into the above catgories, but many do not. Those that do will be mentioned in the appropriate section. Those that do not will be collected under the heading "Miscellaneous".

3.2.1 Transformation Clustering and Hough Methods

The viewpoint consistency constraint implies that all the visible features on a rigid object must be consistent with projection from a single viewpoint. Clustering methods were among the first to employ the constraint, athough Lowe [Low87b] is responsible for its recent appelation. The way that clustering methods use this constraint is by computing the object's feasible viewing transformations and then attempting to locate any clusters in viewing parameter space among the feasible transformations. Feasible transformations are typically computed by forming a set of correspondences between enough image-features and 3-d model-features to yield solution for a unique (or almost unique) set of viewing parameters that consistently projects the 3-d features onto the corresponding image features. Thus, most clustering methods employ object-attached features. Once the feasible sets of viewing parameters have been calculated, objects are recognized by locating clusters of feasible viewing parameters. In such systems, clusters are powerful evidence for the existence of an object in a scene since a cluster indicates transformation that maps many 3-d model features near to image features. The probability of such an event occurring accidentally, especially for large clusters, is small.

Once possible clusters have been identified, recognition is usually based on the properties of the clusters. Typically, the largest or largest few clusters are accepted, or the cluster size is required to be larger than some threshold cluster size.

In a clustering system, computation of feasible sets of viewing parameters is usually the least difficult part of the algorithm. Many possible solutions exist, differing in such details as the number of correspondences necessary to determine the viewing parameters, the number of degrees of freedom allowed in the transformation, and the nature of the features. For example, in the case of weak perspective, where there are six degrees of freedom, [Hut88] gives a solution using three pairs of simple points, whereas a single pair of more complex features called vertex pairs suffices [TM87].

The most problematic aspect of recognition using clustering is the location of *significant* clusters of feasible sets of viewing parameters in a high-dimensional parameter space. Often, the statistical properties of the distribution of the feasible viewing parameters is difficult or impossible to determine, precluding the use of well-understood probabilistic methods. In such cases, various non-probalistic techniques are used. These include variations of the *k-means method*, projection onto lower dimensional subspaces, and the *Hough transform*.

The k-means method is a simple iterative strategy for finding clusters in an n-dimensional vector space. The assumption is made *a priori* that there are exactly k clusters, which is a weakness. The n-d input vectors are divided into k groups, and prototypical values for each of the k classes is computed, often the centroid of the vectors in the class. The vectors are then redistributed to the class whose prototype is nearest, according to some distance metric. This process is iterated until changes in the prototypes are no longer significant.

One of the problems with this approach is that a distance metric must be defined in the parameter space, which often consists of mixtures of rotational and translational parameters. Additionally, proximity in the parameter space may or may not imply similarity

in projected shape.

Another method for finding clusters in high dimensional spaces operates by projecting the feasible points onto lower dimensional subspaces, and searching for clusters there. This is advantageous since finding clusters in lower dimensional spaces is easier than finding them in higher dimensional ones. However, the clusters in lower dimensional spaces could result from the accidental coincidence of points along the dimensions of the projection, and therefore the validity of such clusters should be verified.

The final generic clustering technique that we will discuss is the *Hough transform.* The idea of the Hough transform is to quantize the parameter space into uniform buckets and count the number of feasible parameter vectors in each bucket. Each bucket will tend to contain similar viewing transformations. Buckets with large occupancies will correspond to clusters, which, in turn, indicate likely instances of an object in the image.

There are a number of problems with Hough based methods. This is evident from the analysis in [Hut88]. When the number of "good" features is large compared to the number of "bad" features, i.e., features not resulting from an instance of an object in the image, then Hough methods work well. This is because the peaks in the array of bins is easily located: they are not submerged in a ocean of bad transformations. Unfortunately, in complex images, most of the features do not correspond to instances of the objects being sought. In this case, the probability of large false peaks in the bin array is large, and it becomes difficult to distinguish them from true peaks. One way to combat this problem is to reduce the number of features by making them more complex. Since the features are more complex, there are fewer of them, helping to reduce the number of false peaks. Unfortunately, it also reduces the number of features on the model that can contribute to a true peak in the bin array. In addition, since complex features tend to be less spatially localized, the chance that a feature may be occluded is increased. Further, the size of the bin array becomes astronomical when the dimension of the parameter space is more than four or so. In such cases, projection to lower dimensions becomes a

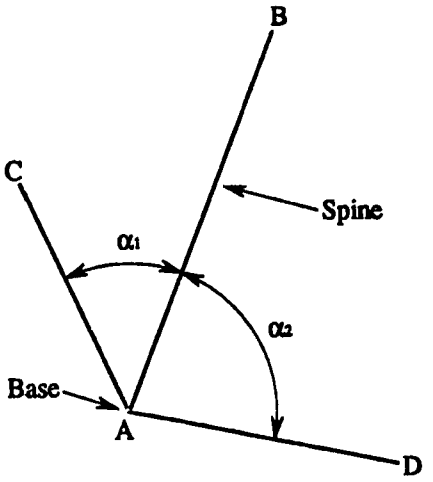


Figure 3.1. A vertex pair feature. AB is the spine, with A as the base vertex and B as the auxiliary vertex.

necessity, which further aggravates the problem of random peaks.

Thompson and Mundy [TM87] provide a good example of using Hough clustering to recognize 3-d polygonal objects in intensity images. Features consist of "vertex-pairs", shown in Fig. 3.1, which come in 2-d and 3-d varieties. The 2-d variety is confined to a plane. Both kinds consist of a "*spine*" which joins two vertex points. One of the vertices is defined to be a "*base vertex*" which has two other edges incident on it, in addition to the spine. The other "vertex" is the point at the other end of the spine. In 3-d, the vertices consist of the vertices of a 3-d polygonal model. The 3-d vertices project to 2-d vertices, and, as shown in Fig. 3.1, the 2-d vertex pair is characterized by the angles α_1 and α_2 between the spine and the edges incident on the base vertex, as well as the spine vector. How such vertex pairs are segmented from the image is not discussed.

For a given 3-d vertex pair, there is a unique weak perspective transformation that projects the 3-d vertex pair to a given 2-d vertex pair. Thompson and Mundy compute this projective transformation using the quaternion technique of [FH83] which allows an algebraic solution. However, to speed the computation of the transformation, the computation is split into in-plane translations and rotations, and out-of-plane rotations. For each 3-d vertex pair in the model polyhedron, the authors compute a table of the

out-of-plane rotation parameters versus the values of the angles in an observed 2-d vertex pair, α_1 and α_2. There is no entry if the 3-d vertex pair is not visible. The set of tables for all possible 3-d vertex pairs comprises the model of each object. Thus, when a 2-d vertex pair is detected in the image, possible out-of-plane rotation parameters can be quickly computed. The remaining in-plane rotation, translation, and scale parameters can be easily determined by aligning the spines and vertices of the 2-d and projected 3-d vertex pairs.

The clustering used by [TM87] is Hough-based, with the 6-d transform parameter space being decomposed into a 2-d space of out-of-plane rotations, a 1-d space of in-plane rotations, and finally, a 3-d scale/translation parameter space. Each correspondence between a 2-d vertex pair and a 3-d model-derived vertex pair yields a set of transformation parameters. Entries are first made in the 2-d out-of-plane rotation bin array, with bins representing two degree increments in α_1 and α_2. The bin array is scanned for peaks indicating clusters, and these peaks are re-histogrammed in a 1-d array for the in-plane rotation. Clusters detected in this 1-d table are further clustered in the 3-d space of translation/scale. This final clustering is done using a variation of the k-means method described above. A cluster in the final histogram with more than three assignments is considered a correct match.

Results were good, although the tests were done on images where the object(s) to be recognized possessed the vast majority of the features. As mentioned earlier, this type of image is the type the clustering algorithms perform best on.

Thompson and Mundy also describe a nearly identical approach to recognition in range images [CMST88]. The Hough-based matching scheme is identical, and most of the novelty resides in segmenting vertex pairs from range images. Strangely, the authors continue to use 2-d vertex pairs even though the range data provides true 3-d features that could be compared directly with the model features rather than though their projections.

Hough-based recognition methods are further beset by problems not mentioned so

far. In particular, choosing the bin size is difficult. Making the bins large reduces storage requirements. Additionally, and more importantly, the chance that the cluster resides wholly in one bin is enhanced, improving the chances of detecting the cluster. On the other hand, since the bins are larger, the chance of large random peaks is larger as well, making detection of true clusters more difficult. Also, since the parameters of the bin are typically used to estimate the pose of the object, a large bin size reduces the accuracy of the estimate of the pose of the object. Making the bin size smaller improves the accuracy of the estimate of the pose but reduces the likelihood that the cluster will fall into a single bin, making detection more difficult. Clearly, if there were no errors in the calculation of the feasible transformations, a very small bin size would be preferable as all the points in the true cluster would fall in a single bin while the chance of large random peaks would be vanishingly small, simplifying detection, and improving the estimate of the pose. However, since there is always error in the transformation parameters, there is an *optimal* bin size that depends on the statistics of the error and the transformations that do not belong to the true cluster.

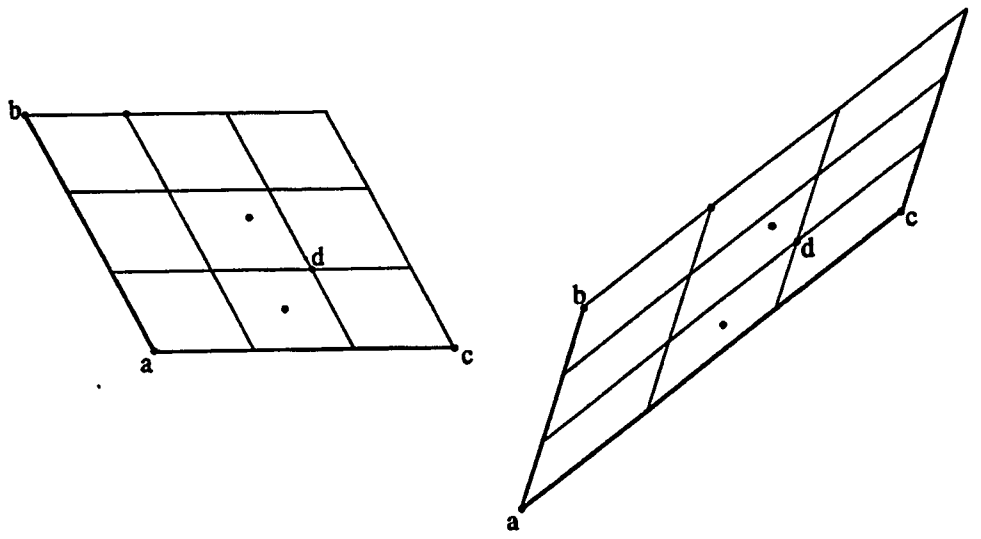
Often, the optimal cluster size for detection is too large for precise pose estimation. In order to overcome this problem, [SDH84, SHD84] discuss an *iteratively subdivided* Hough procedure for finding clusters of feasible transformations in a 5-d parameter space. Detection is done at a large bin size that is good for detection and then the detected clusters are rebinned into smaller an smaller bins until the cluster begins to fragment into multiple bins, providing better pose estimation.

In a similar vein, [LHD88] describes a recognition method that locates clusters in a full 6-d parameter space. Features are 2-d and 3-d triangles. The vertices of the 2-d triangles are generated by intersections of linear contours in the image and the vertices of the 3-d triangles consist of the vertices of the polyhedral model. Corresponding the points of a 2-d and a 3-d triangle leads to a nearly unique solution for a feasible transformation. The feasible transformations are clustered in a 3-d bin array that uses only the translational

parameters, as they can be computed very quickly. Peaks are located by examining a 3×3 neighborhood and suppressing nearby peaks that are likely to be fragmentations of true peaks. Then, peaks in a histogram of translational parameters are detected. Finally, rotation parameters of the transformations are computed and checked for consistency. Visibility of the model triangle pairs can be determined from the rotation parameters, and these constraints are also applied to filter transformations. A further heuristic is applied to determine which of the 3 possible image-triangle to model-triangle vertex correspondences is most likely to be correct. Final acceptance of a cluster is based on how closely the projection of the consistent features in the cluster match with actual image features. Once a cluster has been accepted, a final fit of the model triangles to their corresponding image triangles is done using a least squares measure, yielding good accuracy for the estimate of the pose of the object.

Stockman and Esteva [SE85] describe a similar transformation clustering technique. In this case the transformations are constrained, and result in a 3-d parameter space. Correspondences are formed between pairs of 2-d feature points and 3-d model points, which, in the 3-d transformation space, allow the pose to be determined uniquely. Feasible transformations are computed for all possible pairings of image and model features; unlike [TM87, CMST88] visibility constraints are not applied. Clusters are then detected among the feasible transformations. Although not explicitly stated, it appears that a variant of k-means is used to find clusters. Another method, described in [FHK+82], is very similar.

A series of papers and reports by Lamdan, Wolfson, and Schwartz [LSW88b, LW88b, LSW88a, LW88a] describe an interesting clustering based algorithm for recognizing 3-d objects from intensity images. An earlier paper [KSSS86] describes a similar approach for 2-d objects. At the heart of these methods is a representation scheme that the authors call "geometric hashing". A key component of the "geometric hashing" approach is the existence of a representation of the features that is invariant to the 2-d transformations that the features undergo as the object's pose changed in 3-d space. For example, object-



(a) (b)

Figure 3.2. Demonstration of invariance of Lamdan *et al*'s feature representation to affine transformations. Above, (b) is the result of applying an affine transformation to (a). In both (a) and (b), the feature points **a**, **b**, and **c** form a "basis" coordinate system that has the property that the coordinates are invariant to affine transformations. The other feature points, such as point **d**, can be represented in terms of these invariant coordinates. For example, the coordinates of **d** are (2,1) in both (a) and (b) in spite of the affine transformation between them.

attached features on flat, rigid objects imaged under weak perspective undergo a *affine* transformations as the viewing parameters are varied. If primitive features, such as corner points, are described solely by their position in the image plane (not by additional descriptors such as orientation, curvature, etc.), then a set of three such points defines a *basis set* that forms a coordinate system in which all the remaining feature points can be described. As shown in Fig. 3.2, affine transformation of the points does not affect their coordinates in the basis-set coordinate system. Of course, this is only true if the same basis set is used. Similar representations can be defined for object-attached features on rigid 3-d objects under weak perspective as well; four points are required in this case.

Given existence of such invariant representations of feature points in terms of a multipoint basis-set, the "geometric hashing" approach simply represents *every* feature point in terms of *every* possible basis-set and stores each point's coordinates in terms of each possible basis-set in a hash table data structure. The "hashing" is done by representing each coordinate by a binary number and truncating the low order bits, leading to a discretization of the coordinate space into hypercubes, and then using the resulting binary string as a key into a hash table. The buckets in the hash table are, therefore, representations of various hypercubes in the feature space. The hash table is loaded with all possible representations of all possible features for each model in the system's vocabulary. It is possible that a hypercube may contain more than one feature, especially if the size of the hypercubes is large.

Recognition proceeds by picking a possible basis (three points) in the image and computing the coordinates of all the other feature points in the image in terms of it. Each hypercube that contains an image feature is retrieved. Records, consisting of ¡model, basis¿ pairs, are constructed. For each such record in the hypercube, increment a vote counter for that record. If any particular ¡model, basis¿ record scores a large number of votes, then it a matching candidate.

The correspondence between the image basis and the model basis provides a unique solution to the transformation parameters. Thus, while the transformations are not explicitly represented in this method, they are implicitly coded in the representation. The "voting procedure" is thus voting for a discrete set of possible transformations that are induced by the choice of the image basis points. For this reason the method should be considered to be a transformation clustering approach. The candidate matches and the associated transformations are then retrofitted using least squares and employing any additional, close matching points in the image. Finally, a verification is done between the boundaries of the model and the edge contours in the image.

The method of "geometric hashing" suffers from a few difficulties. First, use of the

term "hashing" is somewhat misleading because it implies constant time access of the records. Since only the hypercubes themselves are indexed in the hash table, strictly speaking, the access time is linear in the average occupancy of the hypercubes. The only way to reduce this is to make the quantization finer. However, this makes it more likely that a noisy feature will not hash to the correct hypercube, similar to the problems faced by Hough-based methods. Under poorer conditions than the high-contrast, backlit scenes reported, the method may break down. Further, extension of this method to true 3-d objects requires that the voting occur at all hypercubes intersected by a line in the feature space, which appears to be an inefficient operation. Finally, the combinatorics of this algorithm are unfavorable since voting must be done for every possible basis set in the image.

There are numerous examples of clustering approaches used to recognize 2-d objects from intensity images [MF75, Bal81, Seg83, BS85, KK85, TMV85, TMV86, Tur86, Hwa87, Ger88, SKB82], a matched dimensionality domain. With one exception [Hwa87], these methods employ Hough-based clustering exclusively. In 2-d methods, the transform space is at most 4-d, consisting of translation, rotation about the normal to the image plane, and 2-d translation within the image plane. In [MF75, Bal81] the pioneering work on applying the Hough transform to the recognition of arbitrary 2-d shapes is described. In common with Hough-based approaches in the 3-d recognition domain, some of these methods attempt to overcome the problem of a high-dimensional parameter space by decomposing the transform bin array into lower dimensional bin arrays [BS85, Seg83]. The approach described in [TMV85] determines weights for each transformation based on a rigorously defined *saliency* measure of the features used to compute the transformation. This greatly reduces the number of random peaks in the bin array, though extending the notion of saliency to 3-d appears to be somewhat difficult. Another method [Ger88] "links" transform space and the feature space by performing, essentially, a *verification* of the clusters detected in transform space by matching predicted features in the image

to filter false clusters from true ones.

3.2.2 The Hypothesize and Verify Paradigm

As we have seen, methods that recognize by clustering attempt to form hypotheses that have considerable global support. Such methods need many features to "vote" before a "consensus" is reached, and, with few exceptions, the clustering is the sole means of accumulating evidence for particular hypotheses. The *hypothesize and verify paradigm* (HVP), on the other hand, is less democratic. In the HVP, only features that can explain the image data in the locality of themselves are allowed to become valid hypotheses. Then, they may win overall by explaining a more global portion of the image data.

Typically, the first step in the hypothesize and verify paradigm is to construct a sparse representation of the image in terms of features. Ideally, such features are highly selective, i.e., model generated features are chosen so that the likelihood of them having the same attributes as an incorrectly matching image feature is very small. Unfortunately, there are practical limits to how selective features can be, as highly unique features tend to be more global in nature, in addition to often being too sparse. Given a reasonable choice of image features, one is chosen by some means, and the set of possible scene instances that could explain its existence, to within measurement error and noise, is computed. Typically, this set is represented by a number of individual scene instances that are treated as separate, competing hypotheses. This process is called the *generation* phase. Next, the existing hypothesis are *verified*. As mentioned in the preceding paragraph, each hypothesis provided by the generation phase is usually a single scene instance. This scene instance is then used make predictions that can be used to perform a detailed check against the image. Based on how well the predictions match the observations, a decision is made as to whether the hypothesis is strong enough to be considered a valid recognition result. Typically, the representations of the predictions and the representations of the observations used in the comparison are more *complete* than the representations

used during the generation phase of the HVP. That is, the representations contain more of the information that is contained in the model instance or the image.

As mentioned in Chapter 1, the exhaustive approach to recognition would be to consider *every* model instance, to within an error resolution volume, as a hypothesis and to verify all of them, passing those that scored high enough in the verification. As was shown in Chapter 1, this is computationally untenable. The hypothesize and verify paradigm overcomes this by considering only those model instances that have at least a small bit of evidence in their favor; usually one, two or a few features that match image features. This filtering vastly improves the efficiency of the search.

In the hypothesize and verify algorithms come in a spectrum of varieties. Aside from differences in the types of features and representations that they use, these algorithms differ in two other respects:

1. how much effort is expended to generate a strong hypothesis, and
2. how features and hypotheses are ranked for further processing.

With respect to item 1, some algorithms have opted to expend the minimal effort generating hypotheses, typically creating many weak hypotheses that are then rejected, while the few strong hypotheses are passed. This approach insures that the correct hypotheses will be very likely to be among the set generated. This approach is robust, but tends to be slow since verification is usually rather expensive in comparison to generation. The other extreme, is to generate hypotheses that are rather strong, and, therefore, likely to be correct. Since the generation algorithm in this approach is discriminatory, it may not allow the correct hypothesis to pass on the basis of the limited information that is has available to it, thus missing a correct interpretation. Also, if taken to extreme, the effort spent generating a strong hypothesis may outweigh the cost of verifying it. Thus, this extreme tends to be less robust, and often just as slow as the other extreme. The optimum tradeoff falls somewhere between the two. [KJ86] describes a 2-d algorithm where the tradeoff is adjusted to minimize the overall time spent generating and verifying hypotheses

under certain assumptions about the way the hypotheses are generated. Unfortunately, such analysis cannot easily be extended to the 3-d domain.

With respect to item 2, the queue of features waiting to be processed can be ranked according to the likelihood that a given feature will generate a strong hypothesis. If the ranking heuristic is good, this will reduce the time it takes to locate objects in the scene. Similarly, hypotheses that have been generated but not yet verified can be ranked in the queue for verification, also reducing recognition time. Most HVP algorithms rank features in some manner. For example, a number of systems use model-independent feature grouping [Low87a, Chi89, Hut88, Jac87] to rank the features. It is less common for systems to rank the generated hypotheses before verifying them; usually they are verified immediately upon being generated. An example of a 2-d object recognition system that does rank hypotheses is Bolles' and Cain's local focus feature (LFF) approach [BC82]. As mentioned previously, they expend considerable effort to generate strong hypotheses. Recall that hypotheses in the LFF approach are graphs of whose nodes are possible image-feature to model-feature pairings, and arcs represent mutually consistent pairings, and whose nodes are fully connected. These hypotheses are then ranked for verification by the size of the graph, which simply measures the number of features that have been matched. Chien and Aggarwal's system for recognizing 3-d objects [Chi89] also ranks the generated hypotheses. In their work, transformations are computed for each hypothesis from hypothesized correspondences between quadruples of image features and model features. The transformation is solved simply as a system of linear equations. In particular, orthogonality of the rotation matrix is not enforced. If the hypothesized correspondence is correct, then the computed transformation should have an orthonormal rotation matrix. The rotation matrix is not likely to be orthonormal if the correspondence is incorrect. To rank the hypotheses, Chien and Aggarwal examine the transformation associated with each hypothesis to determine how close its rotation matrix is to being orthonormal . The hypotheses with transformations containing rotation matrices that are

nearly orthonormal will be processed first, and grossly non-orthonormal cases will be eliminated.

3.2.2.1 3-d Intensity-Based Hypothesize and Verify Methods

An excellent example of the HVP is the ORA (Object Recognition by Alignment) system described in [HU88, Hut88]. This system also typifies how object-attached features may be used to drastically simplify the computation of the hypothetical viewing transformation under the HVP.

The essence of the ORA system is very simple. First, image features, consisting of triplets of points, are put into correspondence with model features, consisting of triplets of 3-d model points. For each correspondence, the ORA system "aligns" the model so that the projection of the triplet of model points comprising the model feature exactly corresponds to the triplet of points in the image feature. Of course, for each pairing of image and model features, there are three resulting possible permutations of image points and model points, and therefore three possible alignments. ORA uses the weak perspective imaging model, and therefore three image-point to model-point correspondences are sufficient to determine the viewing transformation to within a reflection across the image plane. Note that ORA's approach to solving for the transformation is an improvement over that in [Chi89] since the orthonormality of the rotation matrix is maintained. Thus, ORA does not consider any invalid transformations. After alignment, the hypotheses are verified. If a hypothesis is accepted, the image features that have been matched to it are removed from further consideration.

ORA employs curvature-based segmentation of image curves. ORA's creators argue strongly for the use of inflection points, or zeroes of curvature, and line segments as features. They give two reasons for this: first, inflections and line segments are preserved when a space curve is projected, and second, they argue that there is no psychological evidence for high-curvature points over inflection points, citing Lowe's cat [Low85] as

a counterexample to Attneave's [Att54]. Oddly, they go ahead and use high-curvature points anyway. All of the features, are assumed to be object-attached, otherwise the procedure of assuming 3-d to 2-d correspondences and calculating the alignment transform breaks down. As a result of the use of object-attached features, ORA's vocabulary consists of planar or polyhedral objects. The image features are ranked according to how likely they are to be part of the same object in the scene. The heuristics used are:

1. Points are likely to belong to the same object if they appear on the same edge contour, or if they belong to two contours that are likely to have come from the same object.
2. Two contours are likely to be from the same object if their endpoints are in close proximity.
3. Two contours are likely to be from the same object if the relative intensity on either side of each contour is comparable and the contours form a compact geometric shape.

Verification in ORA is a two stage hierarchy: the initial stage is cheap computationally but eliminates many false matches, while the second stage is slower but more accurate. The initial stage operates by checking the endpoints of curve segments in the projection of the model with segment endpoints in the image. Endpoints are either inflection points, high-curvature points, or endpoints of linear segments. Both the position of the points and the direction of the tangents through them are required to be within an error tolerance before an endpoint can be said to match. If more than half of the endpoints visible in the model match with endpoints in the image, then the hypothesis is passed through the initial stage of verification. The detailed verification procedure compares all of the visible contour segments with nearby image segments. If enough of the predicted boundary is near to an image boundary, the hypothesis is passed.

The ORA system is typical of 3-d HVP methods in that it employs an analytic formula to yield the transformation that maps some minimal number of 3-d model points onto 2-d image points. For any such method to work, the features must be object-attached. Some work has been done to extend ORA to smooth 3-d objects [BU88, SU88]. However, it

is unlikely that the alignment method, in its present form, can be extended to recognize smooth 3-d objects.

Lowe's SCERPO system, described in [Low87b, Low87a, Low85], is also a HVP approach. It differs from ORA, which came later, in several respects. First, SCERPO is only able to recognize polyhedral objects since object-attached line segments constitute its feature set. Lowe's work is based heavily on the idea of "perceptual grouping" which is essentially the ability of a vision system to group features based on a model-independent measure of their "perceptual significance". Perceptual significance is really the same as non-accidentalness, i.e., the likelihood that a configuration of features observed is not the result of a visual accident. By making several assumptions about the stochastic properties of the distribution of detected line segments in the image, Lowe is able to come up with an analytic heuristic measuring the perceptual significance of several types of relations between line segments: proximity, parallelism, and colinearity. These significance measures work on pairs of line segments. Significant pairs are further grouped into significant clusters by noting the pairs that share segments.

The "perceptual grouping" process in SCERPO can be carried out equally well on the 3-d model segments as on the 2-d image segments. Hypothesis generation then proceeds by simply matching 3-d groupings to 2-d groupings with the same number of segments, with the groupings with the most segments being processed first since they are the most "significant", although such groupings will lead to a large number of possible permutations in the correspondences of image line segments to model line segments. Image groupings with the same number of segments as model groupings are matched, and the viewing transformation is determined using the correspondences of the individual line segments (three or more uniquely determines the transformation), yielding initial hypotheses. The initial hypotheses are then verified. SCERPO verifies hypotheses by first using the initial hypothesis to find all image line segments that match sufficiently well to line segments predicted by the model. These image-segment to model-segment pairings

are then used to find a least-squares fit between the projected model-segments and the image-segments using a multidimensional Newton-Raphson algorithm. Note that this differs greatly from the AEFMA method developed in this thesis due to the fact that explicit correspondences are made between single pairings of 3-d model line segments and 2-d image line segments. AEFMA maintains a fuzzy degree of correspondence between all possible pairings. In addition, the features in AEFMA are not object-attached. Recently, the fitting technique used by SCERPO has been extended to handle parameterized models [GL87]. Finally, after the least squares fit, the predicted model-segments are again matched to image-segments, and if more than ten matching pairs result, the hypothesis is accepted.

The recent work of Chien and Aggarwal [CA87, Chi89] follows the HVP. Similarly to both the ORA and SCERPO systems described above, Chien and Aggarwal's system detects features in the image (sets of four consecutive "corner-like" features from the edge contours) and forms hypothetical correspondences between these quadruples of image points and quadruples of 3-d model points. This allows the transformation parameters to be solved. As mentioned earlier, Chien and Aggarwal do not enforce the orthonormality of the rotation matrix portion of the transformation, in contrast to ORA, and therefore many inconsistent hypotheses are generated. These are weeded out by applying the orthonormality constraints.

Chien and Aggarwal's algorithm employs 2-d feature points and 3-d model feature points. The 2-d feature points are simply high-curvature points. The 3-d points are chosen by the finding 2-d high-curvature points in three orthogonal "principle views" of the object, and then determining the intersections along the lines of sight from the different views to yield 3-d feature points. Since the 3-d high-curvature points are assumed to project to 2-d high-curvature points, these features are object-attached. The features are ranked by the heuristic that postulates points with higher curvatures as more likely to be reliably detected than those with smaller curvatures. After hypotheses with valid

transformations are found, they are then verified. Verification is a direct comparison of contour shape using a polar representation of the contour with the contour centroid as the origin. This measure does not work for occluded objects. A method for doing verification for occluded objects is discussed, but no results are given. It appears that parts of the hypothesis generation algorithm require knowledge of figure-ground segmentation as well, rendering this algorithm very weak for practical scenes. In its favor, the verification method could easily be fixed, using a method such as the one in ORA, or the one used by Cyclops.

All of the methods described above owe a large debt to the seminal work of Roberts [Rob64], which also followed the HVP. Like SCERPO, Roberts' system used perceptual groupings. In Roberts' system, the groupings consisted of polygons about vertex points in the image. These image polygons were topologically matched to polygons derived from the model. This is also similar to SCERPO's topological matching of perceptual groupings, though SCERPO's implementation is more robust. Following the topological matching, Roberts' system computed hypothetical transformations in a manner similar to the systems described above, though the particulars are most similar to Chien and Aggarwal's system. Roberts's method does not enforce orthogonality of the rotation portion of the transformation, forcing him to patch it up in an *ad hoc* manner. The result of computing the transformation is a set of hypotheses, which are then verified. Verification consists of computing the mean-square error in the projected model points that were members of the original topological match and the corresponding image points.

Other HVP approaches to recognition of 3-d objects from intensity images tend to be very similar to the systems described above, though they may be less complete. For example, the method described in [Whi88] goes through some representational gymnastics in a feature space called "vertex space" to come up with features, called "key features", also known as triangles, that are matched to image features. After matching, a viewpoint hypothesis is generated. As usual, the model is then projected, verification is done by

seeing if enough predicted vertices match observed vertices. Similarly, [PD87] describes a hypothesize and test algorithm that uses a representation called an "asp", consisting of the deformations that the triangles in the polyhedral model undergo as the viewpoint is continuously changed. Features in the image, which are polygons, are compared with the features in the "asp" to see if there are any matches. If there are, a viewpoint hypothesis results. The "asp" is really a form of multiview model. In fact, the features used are object-attached, and therefore, a multiview model is superfluous since viewpoint hypotheses can be computed directly. Details of the verification method are not provided, nor are any results.

A set of two papers by Hansen and Henderson [HH87, HH88] describes a HVP approach called "recognition by strategy trees". The method works with polyhedral objects, and represents them as "strategy trees". A strategy tree consists of a model at its "root". A set of "level one features", and a "corroborating evidence subtree" constitute the body of the tree. The level one features are the "strongest set of view-independent features chosen for their ability to permit rapid identification of an object and its pose". Given an image feature, the matching method finds the model's "level one features" that have similar attributes. These features are used to find the pose of the model, presumably in a manner similar to the other methods described above, or by using visibility constraints. For each such "level one match", a "corroborating evidence subtree" is evaluated. Essentially, this is a verification procedure. Details are sketchy, but the verification seems to check predicted features against observed features and then perform a fit, as in SCERPO. Following that, a detailed boundary check, as in [BC82], is done. The most interesting aspect of this approach is the freedom to tailor the features used in the generation and test modules to optimize performance for each object.

The method of Sato *et al* [STT87] is interesting because it uses a connectionist network both to filter competing hypotheses and to fit the model to the data. Features are based on line segments. Hypotheses are initially generated by pairing "L" junctions

detected in the image data with "L" junctions in the model, and using the hypothetical correspondences to solve for initial viewing parameters. Verification consists of finding "clusters" of compatible hypotheses in the graph of compatibility relations. Each image feature will generate many possible hypotheses seeking to explain it. A "compatibility" measure is defined, and a Hopfield network [HT85] is used to simultaneously adjust the compatibility between hypotheses, yielding clusters of compatible hypotheses. Note that the clusters exist in the graph of compatibility relations, not in viewing parameter space. Each cluster indicates a recognized object. The information in the hypothesese comprising the cluster are then used to refine the estimate of the viewing parameters.

Fisher [Fis83] describes a HVP method that is unique in that it employs *region* based features in contrast to the *edge-based* features used by most other intensity image recognition systems. However, estimating the pose of a model from region data is difficult. Fisher's technique is to correlate the cross sections of model surfaces to segmented image regions, ultimately solved by optimizing a weighted distance measure between the projected model surface and the image region. Hypotheses are represented in a frame-like manner, with slots for transformation parameters, observed surfaces, and instantiated subassemblies. After generation, many of the surface slots and subassembly slots have not been filled. A phase of attempting to fill the slots is entered, implemented by a rule-based system. The initial phase, described earlier in the paragraph, can be called to instantiate subassemblies as necessary. Once all slots are filled, the complete hypothesis is verified. Verification consists of reestimating transformation parameters, checking if projected model surfaces substantially cover the image regions assigned to them, and finally, checking if the predicted boundary matches well against the observed boundary.

The primary weaknesses in Fisher's algorithm include the method for estimating the initial viewing parameters using correspondences between model surfaces and image regions, and the requirement that the segmentation of the image be very good. Hand segmented images were used in to obtain the results shown in the paper. Strengths of the

algorithm include its use of hierarchical models and hierarchical matching. This helps to improve the efficiency and robustness of the algorithm.

3.2.2.2 MDD Hypothesize and Verify Methods

Many 2-d recognition systems have been reported that use the HVP. Some of these methods exploit the symmetry that exists between the model domain and the image domain, with the result that they are difficult or impossible to extend to the 3-d case. A example of such a method is the well-known "local-feature-focus method" (LFF method) [BC82]. This method was designed on the premise that verification is very expensive, so generating good hypotheses (i.e., ones likely to be correct) is important. Hypothesis generation is done via a graph search in a graph where the nodes represent possible image-feature to model-feature pairings, and arcs between nodes represent compatible pairings. Features are circles and corners. Roughly speaking, consistency is determined by the absence of competition for the same features, as well as agreement of relative poses between image features and model features as represented by the pair of nodes in the graph. The graph is the searched for the largest sets of mutually consistent pairings of image features and model features. In order to improve the effiency of the NP graph search, nodes which contain hand-selected "focus features" are used to focus the search for maximally connected subgraphs. The largest such set is used to determine the viewing transform, a rotation and translation, by aligning corresponding features. This hypothesis is passed to the verification module, which directly compares the transformed model boundaries to the image boundaries using probes that are perpendicular to the model boundary. Dark-to-light transitions are positive evidence, light-to-dark and all-light transitions are negative evidence, and all dark transitions are neutral. This scheme assumes that the relative brightness of objects and background is known *a priori.*

A method that is similar in many respects to the LFF method called "3DPO" is described by Bolles *et al* in [BHH83, BH86] 3DPO attempts to recognize 3-d objects in

range images. 3DPO's hypothesis generation apparatus is very similar to that of the LFF method described in the preceding paragraph. The features differ, being based on range discontinuities. They include circular arcs and linear segments, along with information about the surfaces on either side of them. As in LFF, the modeling system allows the user to label certain features as being particularly selective, as in the "focus features" of LFF. The viewing parameters are computed by successively constraining the possible transformation parameters as each image feature is matched to a model feature. In this regard, the hypothesis generation portion of 3DPO is an instance of the predict-observe-backproject recognition paradigm discussed in the following section. Verification consists of a comparison between the predicted range image that the model would produce with the actual data.

Rearick *et al* [RFC88] describe a method based on a connectionist network (as distinct from a neural net) that, they argue, is fundmentally different from any "model-based" method. The method is region based, and classifies regions into three types: *hons* (Japanese for long, thin objects like pencils) , *cusps, and loops*. The regions are detected using a modified medial axis transform [Hea86] and classified according to the topology of the skeleton and the width of the region about the skeleton. Relationships between the regions are used to recognize objects. Each object has a customized network that "filters" out sets of regions whose relations are not sufficiently similar to corresponding model relations. In fact, the network is really doing nothing more than a clique-finding procedure similar to that hypothesis generation portion of the LFF method, one of the methods that Rearick *et al* say they are so different from. There is no explicit verification procedure.

Part of the work in this thesis is based on work described in [GTM89, GTM87]. While that work will be discussed more completely in the coming chapters, a brief summary is included here for completeness. The method is for recognizing 2-d objects, but the hypothesis generation portion of it has been incorporated into Cyclops with few

modifications. The features, called "CPN's", are vectors that efficiently encode the shape of edge boundaries in the neighborhood of high-curvature points. Using training images, models of objects are constructed that consist of both the sparse CPN representation and a complete x, y, and slope-angle (θ) versus arclength representation. The models are stored in a special vector-associative memory, implemented with a k-d tree [Ben75] indexed by the five descriptive parameters of the CPN's. The associative memory allows a model containing a feature that matches an image feature to within a user-specifiable tolerance to be retrieved in $O(\log N)$ time, where N is the total number of model CPN's stored in the memory. To the author's knowledge, this is the most efficient hypothesis generation method in the literature most of which are linear at best. Hypotheses are generated by querying the associative memory for models possessing features that are similar to the image feature, and then aligning the model feature with the image feature. These hypotheses are then verified in a hierarchical fashion. First the percentage of CPN features predicted are compared to those detected to reject many possible hypotheses. Then a detailed boundary check similar to, but more general than, the one used by the LFF method is performed.

The GROPER system [Jac87] uses a hash table to implement an associative memory for hypothesis generation. The indexing is done on quantized versions of five parameters that describe the geometric relationship between two pairs of line segments. This approach has the usual problems associated with quantizing the parameter space (see the discussion of clustering based methods, and in particular, [LSW88b, LW88b, LSW88a, LW88a] above). Verification consists of checking how many detected line segments are close to predicted line segments. GROPER is not unique because of its generate and test modules, but rather in how it uses model-independent grouping to reduce the number of hypotheses that are generated. Edge grouping is done on the basis of relative proximity and orientation similarity. The number of hypotheses generated was reduced by nearly a factor of 400 by the inclusion of the grouping module. An improvement in accuracy

was also noted, primarily because the verification module in GROPER is weak, and the relatively powerful grouping module assisted the verification module.

Perkins [Per77, Per78] represents both the image boundaries and the model boundary in terms of "concurves", sequences of line segments and circular arcs. Concurve sequences are matched in a correlational manner. If a model has a subsequence of concurves that matches a subsequence of an image concurve, then a transformation is determined, and a hypothesis is created. The transformation is determined by aligning the matching concurves in tangent-slope-angle versus arclength space, and the hypothesis is checked in a manner similar to the LFF method, differing only in that the presence of an edge pixel with the proper direction is positive evidence, and there is no negative evidence. This is superior to the LFF verification technique as it requires no *a priori* assumptions about the illumination and reflectance of objects relative to the background.

Methods that rely on correlation of the portions of the model boundary with the image boundary for matching or pose determination, as in Perkins method above, cannot easily be extended to the general 3-d recognition. This is because the shape of the model boundary may change drastically with viewpoint, requiring that a correlation be performed for many points viewing parameter space. Since correlation is often time consuming, performing a correlation for a large number of viewpoints would be very slow. In effect, the boundary representation is too complete to allow efficient hypothesis generation.

Another method that uses correllation to generate hypotheses is described by Knoll and Jain in [KJ86, KJ87]. In this example, portions of the model boundary are correlated in Cartesian space. If the segments match well enough, the translation and rotation necessary to align the features are determined, and a hypothesis is created. Verification is nearly identical to that in the LFF method. What is unique about this algorithm is that the features are chosen to minimize the total recognition time under the assumption that the model features and the image boundary is searched in a linear fashion. Other correlational

approaches to 2-d recognition are described in [Fre77, DKZ79, YMA80, BBR83].

Ettinger [Ett88] extends the work of Knoll and Jain in the direction of improving recognition time. He describes how employing a sub-part hierarchy can markedly improve recognition time complexity. The features employed by his system are those of the "curvature primal sketch" [AB86]. The features are contained in a scale hierarchy where the coarsest level of features is used to generate the initial hypotheses about the subparts of an object. Subpart hypotheses that have enough support in the form of more matching features at finer resolutions of the scale hierarchy can generate full object hypotheses. These, in turn, can direct the search for other subparts. The full object hypotheses are hierarchically verified through the subparts.

The method described in [TFF88] is a HVP method that has been designed with parallel implementation on the Connection Machine [Hil87] in mind. Features are line segments and corners (corners are line segments whose endpoints are near to their intersection). Since the transformation space is 3-d, each image-corner to model-corner correspondences allow a model transformation to be computed, and a hypothesis to be generated. Since all hypotheses are generated in parallel, an initial level of verification is performed by clustering the hypotheses in the model transform parameter space. Clusters indicate hypotheses with a large degree of mutual support. A multiscale Hough-like method is used to do the clustering. From the clusters, an aggregate transformation is determined, and a refined hypothesis is generated. Verification consists of comparing transformed model line segments with image segments.

3.2.3 Predict-Observe-Backproject Paradigm

As its name implies algorithms based on the predict-observe-backproject paradigm, or POBP, has three basic steps that are iterated. First, *predictions* are made about what may be observed in the image. The predictions are based on the models and the current state of the algorithm. Typically, the type, attributes, and pose of features are predicted.

Next, the image is examined for *observed* data that match closely with the predictions. Assuming such a match is found, the implications of these matches are "backprojected" into the space of scene instances, resulting in one of two outcomes: either the scope of feasible scene instances is narrowed by the additional constraints induced by the match with the current set of feasible scene instances, or, the induced constraints result in an inconsistent solution. When an inconsistent solution is generated, POPB algorithms typically backtrack to a previously visited feasible solution, i.e., one containing a non-empty set of scene instances. This loop is iterated, usually resulting in the tree-structured search that is the hallmark of the POBP.

3.2.3.1 Backprojection

The heart of the POBP is *backprojection*, the propagation of constraints induced by a match between a predicted feature and an observed feature. Strictly speaking, any algorithm that uses a match between model features and image features to calculate a viewpoint is doing a form of backprojection, though not necessarily in its fullest sense. Therefore, all transformation clustering approaches and many HVP methods employ a limited form of backprojection. However, backprojection in its full sense requires a probability distribution on the space of scene instances. For example, suppose that an image feature has been assigned to match a predicted model feature. Assume that these features consist of triplets of primitive feature points. Further, for simplicity, assume that the features can be considered to be object-attached. Therefore, the features each possess six attributes, specifically, the x and y coordinates of each of the three primitive feature points comprising each compound feature. Thus, under weak perspective, a particular correspondence between the image feature and the object feature will yield a unique solution, up to reflection though the image plane, for the pose of the model that causes the predicted features to coincide with the observed image features. That is, if the observed feature's attributes are known with full certainty, then a point in the observed

feature's attribute space maps to two points in the scene instance space, as shown in Fig. 3.3. In reality, however, a feature's attributes are never known with certainty due to noise and various other distortions. Rather, there is a probability distribution on the feature attributes. Through the mapping from feature attribute space to scene instance space, a probability distribution on the space of scene instances is induced. Denote this probability distribution as $P(s|\mathbf{f}^i \equiv \mathbf{f}^m)$, where s is a scene instance (which may be the null instance), $\mathbf{f}^i$ is the image feature, and $\mathbf{f}^m$ is the predicted model feature, and $\equiv$ indicates that $\mathbf{f}^i$ matches $\mathbf{f}^m$. Figure 3.4 illustrates the probabilistic definition of backprojection. This definition can be carried further to the case of n matching image and model features. In this case, the distribution is

$$P(s \mid \mathbf{f}^i_{j_1} \equiv \mathbf{f}^m_{k_1}, \mathbf{f}^i_{j_2} \equiv \mathbf{f}^m_{k_2}, \ldots, \mathbf{f}^i_{j_n} \equiv \mathbf{f}^m_{k_n}). \tag{3.1}$$

Explicitly calculating these conditional probability distributions has never been attempted owing both to the inherent intractability of the problem as well as to the fact that the distribution is dependent both on the details of the model and types of features used. What is typically done is to divide scene instance space into regions $\mathcal{R}$ and $\neg\mathcal{R}$ such that $P(s \in \neg\mathcal{R} \mid \mathbf{f}^i_{j_n} \equiv \mathbf{f}^m_{k_n}) < \epsilon$, and work with these regions rather than the probability distributions themselves. Figure 3.5 shows an example of such regions. Following Cass [Cas88a, Cas88b], in the following paragraph, we will refer to a region $\mathcal{R}$ as a *match region* because all scene instances $s \in \mathcal{R}$ project $\mathbf{f}^m$ to within a neighborhood of $\mathbf{f}^i$ defined such that the undistorted value of $\mathbf{f}^i$ falls in the neighborhood with probability $1 - \epsilon$.

Backprojection alone is the basis of only one recognition algorithm that we know of [Cas88a, Cas88b]. This algorithm bears a superficial similarity to the transformation clustering based methods discussed previously, but actually differs in important respects. The work addresses 2-d recognition: rigid image-plane rotations and translations of models are allowed. The viewing parameter space for this method is therefore three dimensional. However, the idea is general, and could be easily extended to recognition of 3-d objects from 2-d images using appropriately chosen object-attached features.

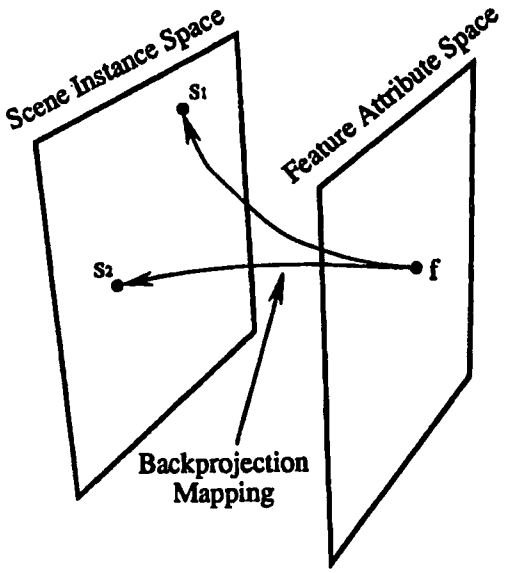


Figure 3.3. An illustration of noiseless backprojection. The vector of measured attributes of a projected model feature is **f**. In this example weak perspective is assumed, and, as explained in the text, features consist of triples of 2-d image points and 3-d model points. In this case, perfect, noiseless measurements are assumed. Thus, measured values for **f** imply that the model can have two possible poses. Therefore, **f** *backprojects* to two points s_1 and s_2 in scene instance space, as shown schematically.

Features in this algorithm consist of evenly spaced points on an image or model boundary, along with the orientation at each point. Due to the isomorphism between the image domain and the model domain, the representation of model and image boundaries and their derived features are identical.

For each correspondence between an image feature and a model feature , a match region is defined; i.e., regions where the projection of an of a model feature will result in a predicted feature whose attributes are within a neighborhood of an observed feature. The size of the neighborhood models the amount of distortion caused by the imaging process. In [Cas88a, Cas88b], match regions are simply cylinders in the 3-d viewing parameter space. The dimensions of the cylinder are chosen so that the transformed feature lies within a neighborhood of the undistorted image feature with high probability. Consider,

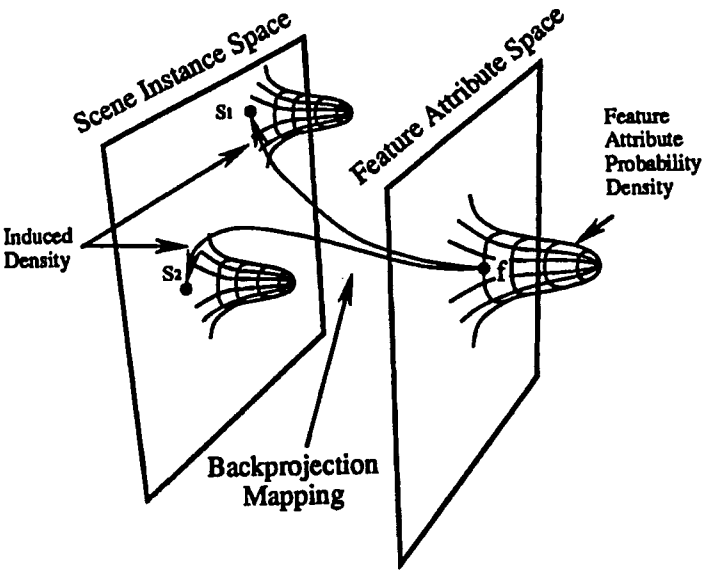


Figure 3.4. An illustration of backprojection in the presence of measurement error. As in the case of Fig. 3.3 above, the vector of measured attributes of a projected model feature is **f**. In this case, the measurement is not assumed to be perfect, i.e., there is a probability distribution on **f**. Backprojection of this distribution then induces a probability distribution on the space of scene instances.

for every pairing of an image feature with a model feature for a particular model, the intersections of all of the match regions. Let every non-null intersection of two or more match regions be called an "intersection volume". Each intersection volume constitutes a hypothesis that the model appears at a transformation contained within the intersection volume with probability $1 - \epsilon$. Clearly, the best such volumes are likely to be those that are the result of the intersection of a large number of match regions, as such intersections volumes are comprised of the transformations that place a large number of features near to the true image feature with high probability. For each model, the intersection volumes that are comprised of the intersection of greater than a certain number of match regions are the hypotheses considered to be valid recognition results.

This algorithm is simple, elegant, and highly parallelizable, as was demonstrated by its implementation on a Thinking Machines Corp CM-1 Connection Machine [Hil87]. Match regions were represented by a uniformly sampled grid of points inside each region

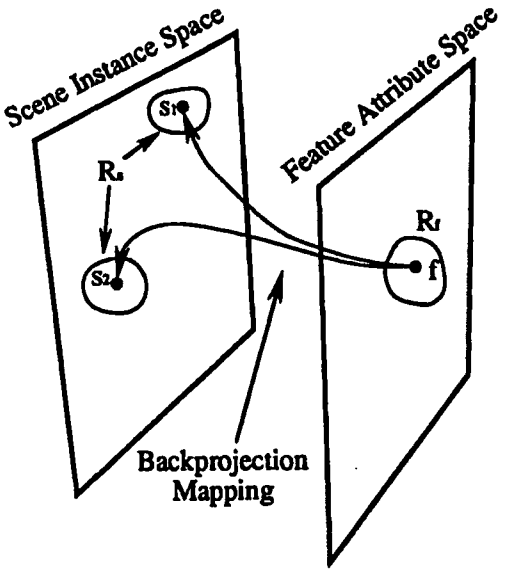


Figure 3.5. An illustration of a common approximation to full backprojection. The situation is as in Figs. 3.3 and 3.4. Typically, full back projection, as illustrated in Fig. 3.4, is replaced by a simple approximation, as shown above. Region R_f, containing most of the probability density of f, backprojects to region R_s, which, in turn, contains most of the induced probability density. Most methods that perform backprojection work with such regions, or further approximations to them.

so that the intersections could be accomplished efficiently in parallel. A more refined version of the method, also described in [Cas88a], uses approximations to the conditional probability distributions discussed previously to give even more accurate results.

3.2.3.2 The Predict Observe and Backproject Paradigm

Having examined backprojection, the heart of the POBP, in some detail, we can now complete describing the POBP.

Prediction can also be viewed as *forward projection.* Given a probability distribution on the error in an observed feature, backprojection induces a probability distribution on the set of scene instances. In contrast, in the case of prediction, or forward projection, the probability distribution on the set of scene instances induces a distribution on the

attributes of a feature. This probability distribution can be used to match observations during the *observe* step of the cycle. That is, it allows us to assess the probability that a predicted feature falls within a neighborhood of an observed feature in feature-attribute space. If the probability is large, then the POBP may assign the predicted feature to match the observed feature. The implications of this match are then found by backprojecting the measurement error distribution of the observed feature by recomputing the conditional probability given in (3.1). Algorithms employing the POBP typically maintain a measure of the current goodness of the solution. One measure is the volume, in scene instance space, for a given model, that contains a sizable portion of the conditional probability distribution. If the volume of such a region goes to zero, the solution can be considered inconsistent, and backtracking is usually prescribed. If, on the other hand, a small region containing much of the conditional probability exists, it is likely to be correct. Thus, the POBP usually takes the form of a tree search, with each node representing an additional hypothesized match between a predicted feature and an observed feature. If the measure of the goodness of the solution increases above a threshold, then an object is recognized.

As mentioned previously, backprojecting the implications of a feature match is difficult for the case of 2-d data and 3-d objects. This may explain why there are few reported methods for using the POBP to recognize 3-d objects in 2-d images. In matched dimensionality domains, by contrast, full-fledged backprojection can be approximated in such a way that it becomes trivial. Thus, there are a larger number of tree structured methods in matched dimension domains that fall into the category of POBP than in general domains. However, the backprojection portions of these algorithms are somewhat atrophied. These will be mentioned in later paragraphs.

Perhaps the two best examples of algorithms that employs the POBP for recognizing 3-d objects in 2-d intensity images are the methods of Goad [Goa83] and ACRONYM [BGB79, Bro81, Bro83, CCL84]. We feel that Goad's approach is better overall. In particular, Goad's method is actually has been shown to recognize 3-d objects whereas

ACRONYM has never been shown to work on true 3-d scenes. Further, ACRONYM, while it has much to contribute, has serious flaws that would prevent it from ever becoming a practical system. The problems with Goad's approach are of a more subtle nature, problems that Cyclops has been designed to overcome. Therefore, we will describe Goad's system as the archetype of the POBP.

The features used by Goad's approach are line segments. 2-d line segments are assumed to be the result of the projection of one of the edges of one of the polyhedral models. Thus, the method uses object-attached features.

Models consist of a list of the 3-d edge segments comprising a polyhedron. Each 3-d model segment has an associated list of facets on a partition of the viewing sphere from which it is visible, called a "visibility locus". The viewing sphere is partitioned into 218 view regions. The locii are represented as bit strings that denote the union of some of the 218 view regions. Each visibility locus for each model is precomputed since the locus does not change during recognition.

At any time during the course of a solution, Goad's system maintains a current "locus of visibility", denoted L, which is the current set of viewpoints that could account for the visibility of the currently matched image and model line segments. The locus L is Goad's approximation to the conditional probability density described above.

At the start of the algorithm, all possible model edges are considered candidates to match any model edge. Once the initial match is made, the visibility locus L is initialized to the visibility locus of the model feature (the first backprojection). Prediction is accomplished in two steps. First, an unassigned model edge whose visibility locus has a non-null intersection with L is selected. The second part consists of two cases: (1) assume the edge is visible; and (2) assume that the edge is actually invisible. At first glance, this may seem to be contradictory since, if L intersects the visibility locus of the currently selected model edge, then it should be visible. Actually, this is not so, since the true viewpoint of the model may lie inside a portion of L that lies *outside* of

the intersection of L with the visibility locus of the edge. This situation is illustrated in Fig. 3.6. In the case where the edge is assumed to be visible, L is updated to be the intersection of itself with the visibility locus of the currently selected segment. The position and orientation of the projection of the model edge relative to the projection of the initially matched model edge is then computed as the viewpoint ranges over the current visibility locus. The range of relative locations and orientations, plus the known location of the first edge (since it has already been matched), and some account for measurement error, provides bounds on the location and orientation of image edges that could match the projection of the model edge. If, after attempting to extend the match further, the algorithm finds that the current match is inconsistent with the assumption that the current model segment is visible then the algorithm assumes that the current segment must be invisible after all. It then restores L to its state before the visibility assumption was made, and then updates L to be the intersection of itself with the *complement* of the visibility locus of the currently selected segment. If this intersection is empty, the algorithm backtracks to a previous choice point. If there is an intersection, the algorithm chooses a new model segment and continues with a new prediction as in the visible case.

Observation is simply the process of checking the list of detected image segments for those whose position and orientation fall into the bounds predicted for the model edge we want to match. If any such image features exist, extend the hypothesis to include one of them as a match.

Backprojection refines the visibility locus of the current hypothesis, L, by restricting it to a smaller locus, say L', that is consistent with the measured pose of the image feature that was matched to the currently selected model feature. This is done by computing the pose of the current model feature at all points in L and retaining those that result in the projection of the model feature having the same relative pose to the initially matched edge as measured from the image segment.

A hypothesis is accepted if its "reliability" is greater than threshold, and backtracking

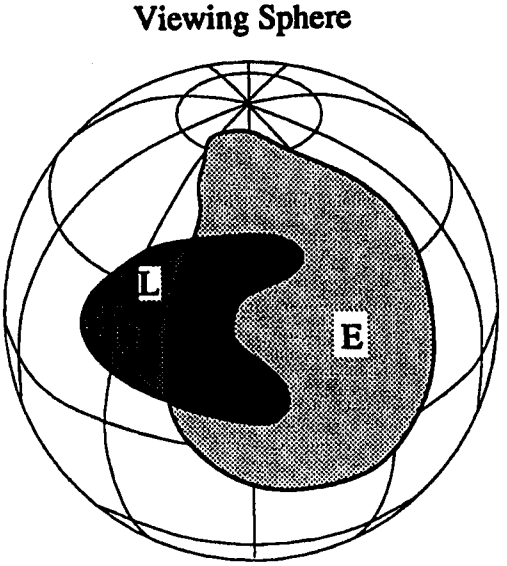


Figure 3.6. L is the visibility locus of the current solution, and E is the visibility locus of the model segment currently under consideration as a possible addition to the hypothesis. The dot in L denotes the correct viewpoint, under the assumption that the hypothesis so far is correct. Although E has a non-null intersection with L (the dark gray region), the edge is actually *not* visible since E does not contain the true viewpoint.

occurs if L becomes null or if the "plausibility" of the match falls below another threshold. "Reliability" is similar to "perceptual significance" as defined by Lowe [Low87a], and is thus synonymous with "non-accidentalness". Details on the calculation of the reliability measure are not given. However, it appears to be done in a manner very similar to Lowe's method. "Plausibility" is a measure of the likelihood that the edge detector would have missed the edges that were predicted to be visible in the current hypothesis. However, poor edge detection is not the only way that edges that are visible can fail to be detected; occlusion is another possibility. The algorithm does not take this into account, and thus is restricted to recognizing largely visible objects.

Since the relationships between the features are all relative angular relationships, the values of the viewing parameters not described by L, which include image-plane translation, rotation, and scaling, can be determined by examining any pair of edges. No

fitting is done to improve the final estimate of the viewing parameters.

ACRONYM [BGB79, Bro81, Bro83, CCL84], an ambitious system that has been cited often in machine vision literature, also follows the POBP. The system is complex, and its description will be heavily abridged in this discussion. Objects are modeled via an "object graph" which consists of two subgraphs. The first is a subpart graph whose nodes are assemblies, and whose arcs are directed from complex to simple assemblies. The other subgraph, called the "restriction graph", allows generic classes to be represented by placing bounds on the dimensions, number, and relative poses of subassemblies. These two subgraphs are actually trees. The leaves of the trees are modeling primitives called "generalized cylinders". Generalized cylinders (GC's) are specified by a "spine" and a "sweeping rule". A GC is created by sweeping a cross-section, specified by the sweeping rule, along the spine of the GC. In the general case, the cross-section may change arbitrarily along the length of the spine, while remaining perpendicular to it. Thus, GC's are very flexible modeling elements. However, ACRONYM actually uses a small subclass of all possible GC's. ACRONYM allows GC's that have rectangular, hexagonal or circular cross-sections. Spines may be linear or circular; GC's with linear spines are allowed to have their dimensions linearly varied along the length of the spine, while those with circular spines must have constant cross-sections.

Perspective projection of ACRONYM's restricted class of GC's results in 2-d ellipses, trapezoids, and hexagons. The features that ACRONYM uses reflects this: ellipses and "ribbons" constitute ACRONYM's feature set. A ribbon is the 2-d analog of a a generalized cylinder. Ribbons are restricted to have linear spines and sweeping rules, implying that they are actually trapezoids. The trapezoids and ellipses are found by applying an edge detector then a linker followed by a line finder [NB80]. Ribbons and ellipses are detected from the line segment data, and become the nodes of the "observation graph". Arcs of the observation graph are relationships between features. Only connectivity between ribbons was implemented. The matching in ACRONYM is based entirely on this

ellipse-ribbon level of description.

Matching follows the POBP. At first, there are few, if any, constraints on the poses of the objects and their component subassemblies. Thus, the possible poses and shape attributes of the ellipses and trapezoids that could result from the projection of any of the object graphs vary widely, allowing many possible matches between predicted features and observed features. Later in the interpretation process, the constraints on the poses of the objects and their subassemblies reduce the variance in the attributes of the predicted features, reducing the number of observed features that are likely to match, as in Goad's method above. Predictions are placed in a "prediction graph" whose nodes consist of *features* and *bounds* on their attributes, or, recursively, other prediction graphs. Arcs describe relationships between the features, such as relative spine orientation and connectedness.

Observation consists of querying the observation graph for any features that match primitives in the prediction graph and that are consistent with any observed-feature to predicted-feature matches already in effect.

If an observed feature is assigned to match a predicted feature, then the attributes of the predicted feature (which may be quite loosely specified) are set equal to the observed feature. The implications of this match are *backprojected* in the form of tighter constraints on the pose of the objects and their subassemblies, reducing the possible ranges of the viewing parameters and the model parameters. The manner in which this is done is at the heart of ACRONYM, and, in this author's estimation, is one of the primary reasons that ACRONYM never worked on true 3-d images. All transformations are represented symbolically by products of primitive transformations, which, in turn, are parameterized by various angles and displacements. The forward projection and backprojection mappings are, therefore, described by very complex, coupled sets of symbolic trigonometric equations. A symbolic algebra manipulation system was employed to propagate the effects of fixing a set of feature attributes back into the object graph. The same system

was used to propagate the new bounds on the attributes of the predicted features that result from the tightening of the backprojected constraints. The problem is that solving such systems of equations exactly is intractable, so the system was forced to make liberal approximations. The approximations could be so bad that the resulting upper and lower bounds are useless in constraining the search.

While a great deal of effort was put into ACRONYM to handle general 3-d scenes, none of the results published have ever shown ACRONYM is able to recognize general 3-d objects. Most results show interpretation of aerial scenes from a fixed viewpoint, a problem that could be solved much more simply. Extensions were shown for a set of switch parts [CCL84], however, only stable poses were allowed. These situations reduce to 2-d recognition. Further, the use of a restricted set of GC's hurt the system, as well as staying at or above the level of ribbons and ellipsed in the level of data abstraction, preventing lower level information from strengthening or weakening interpretations.

Another POBP method for recognizing 3-d objects in 2-d intensity data is described in [BAM86]. Features in this method are corners and line segments. Large "blobs", or regions enclosed by these features, are also found. The search is tree-structured through the space of possible model-feature to observed feature pairings. Backprojection consists of using either a least-squares method or a method based on the ratios of the areas of blobs to find the transformation that best maps the model features onto the observed features. No effort is made to approximate the conditional probability density; a single point in transformation space is found, with the implicit assumption that some neighborhood of this point is also valid. The tree search is guided by an admissible heuristic that encodes information about the geometric mismatch between predicted features and observed features, as well as noise in the feature detection process. Results are given for images of wholly visible fighter planes.

There are two probable reasons for the relative scarcity of methods that use the POBP to recognize 3-d objects in 2-d intensity images. One is that performing backprojection is

very difficult in all but the most simplified cases, even when object-attached features are employed. The second is that, empirically, it appears that reliable recognition of 3-d objects in 2-d intensity images requires that, at some level, the prediction be very *complete*. In contrast, most POBP-based methods employ a sparse feature-based representation of the predictions of the objects. They do this primarily to cut the combinatoric complexity, although the complexity of performing a tree search in an inhomogeneous feature space is also likely to be a factor. The incremental verification scheme of Cyclops is directed toward handling this problem, while retaining robust performance.

3.2.3.3 Matched Dimension Domains and the Predict Observe Backproject Paradigm

POBP-based methods that operate in matched dimension domains are more plentiful than methods that operate in general domains. This is partly because backprojection becomes simpler, amounting to nothing more than alignment of model features with observed image features to within measurement errors. In addition, less complete representations of 2-d predictions can be used and still result in a a robust approach, allowing a tree search in a homogeneous feature space to be used.

There are several well-known examples of such approaches. One of these is the method described in the following papers by Grimson and Lozano-Perez [Gri88a, GLP84, GLP85, GLP87]. This method is applied both to recognition of 3-d objects from sparse range data, as well as recognition of 2-d objects in 2-d intensity images. The approach for recognizing 3-d objects in range data 3-d approach is described; the 2-d case recognition case is completely analogous. Both models and images are represented in terms of their features, planar patches. The method searches an "interpretation tree" (IT), whose nodes consist of all possible strings of observed-feature to model-feature pairings. The length of the strings corresponds to the depth of the node in the tree. Observed features may also be paired with a "null face", indicating that the feature is spurious with respect to

the current model. The pairings are first "filtered" by clustering the model-feature to image-feature pairs in a subspace of the full 6-d viewing parameter space using a Hough approach. Branches of the IT that have pairings that belong to the same cluster are regarded as more likely to represent valid interpretations. Early versions of the method [GLP84] used decoupled geometric constraints between pairs of observed features and model features to prune the IT, i.e., a branch is pruned if the latest set of two pairs of model features and observed features do not have the same relative geometric relationship, to within measurement error. Thus, the search never explicitly worked in viewing parameter space, and so there was no backprojection step. More recently, however, the authors used full coupled constraints that consist of determining the viewing parameters based on the features matched so far. This constitutes a simple form of backprojection, to prune the IT. If backprojection indicates that there is no set of viewing parameters that can map the model faces onto the observed patches to within measurement tolerances, then the branch is pruned. The search of the IT proceeds in depth-first fashion until the interpretation becomes valid, or is pruned. An interpretation is considered valid if it explains a large enough portion of the area of the range image.

This method has been extended to work with curved objects in 2-d [Gri89, Gri88c] and parameterized 2-d objects [Gri88b]. There are a number of MDD methods that are very similar to the one described above. They include [Gre86, KK87, FH83, AFF84, AFFT85, AF86, MC88, PILSL88].

An interesting algorithm is described by Van Hove in [Van87a]. This method is very similar to the early version of Grimson and Lozano-Perez's algorithm wherein the branches of the tree search were pruned using pairwise geometric relations between features. The novel part of Van Hove's approach is that this idea is used for 3-d recognition from 2-d intensity images. This is suprising because such geometric relations are highly dependent on the viewpoint, and may take on a wide range of values over the viewing sphere. Such wide ranges on values of the attributes of the geometric relations reduce

their pruning power. Grimson and Lozano-Perez's method needed only to account for measurement error, which it typically much smaller in magnitude than the variations due to changes in viewpoint. Since Van Hove's method is very similar to Grimson's method, and since it performs no backprojection during its tree search phase, it is discussed here rather than earlier in this section.

Van Hove's method employs pairs of linear edge fragments as features. The key to the method is a preprocessing step where each model is rotated to all possible views in a densely populated sampling of the viewing sphere. The range over which each feature's attributes vary is recorded and stored as part of the model. The space of image-feature to model-feature pairings is searched, as in Grimson and Loazano-Perez, without the use of Hough clustering as a heuristic. At each node, a model feature is allowed to match an image feature only if the image feature's attributes are within the allowable precomputed bounds of the model feature's attributes. If not, the branch is pruned.

Were the tree search the only means of finding valid interpretations, it is likely that Van Hove's method would not work well. As it is, the tree search is used only to *generate* good hypotheses for testing, somewhat in the spirit of Bolles and Cain [BC82]. During the hypothesis test, the features matched by the tree search algorithm are used to compute an estimate of the viewing parameters of the model. These, in turn, are used to generate a more complete prediction of the appearance of the model, which is then compared to the image to decide which hypotheses represent valid interpretations.

3.2.4 Global Feature Methods

A *global* feature is a feature whose attributes are calculated based on the entire region that an object occupies in an image. Typically, global features are *shape descriptors*, i.e., their attributes encode the *shape* of the region that the object occupies. Usually they are constructed to be invariant to image-plane translations, rotations, and scalings. Thus, ideally, their attributes change only when the shape of the object changes. Under weak

perspective, the usual imaging assumption for such methods, this can happen only when the viewpoint changes.

By their nature, methods that rely on global features assume that an object has been accurately segmented from the background, leaving only the problem of *identification.* In certain domains, this is a reasonable assumption. In most domains, however, such segmentation is very difficult. Because of this, global feature methods are of little practical value. However, in their favor, unlike all other methods that have been examined so far, these methods do not employ the object-attached feature assumption. This follows because the region over which a global feature is calculated is the silhouette of the object, and, as shown earlier, any feature depending on the shape of the silhouette cannot be object-attached.

Since global features are not object-attached, analytic methods for determining the viewing parameters of the model given the values of the feature's attributes do not exist. Therefore, the mapping from feature attributes to viewpoints must be precomputed and stored in a form of *multiview model.* Indeed, it is a trademark of global feature methods that they employ some type of multiview model.

There are many types of global features. Some common examples are area, perimeter, compactness (ratio of area to perimeter squared), Fourier descriptors [PF77, Gra72] and Wigner distributions [JW84], and moment invariants [Hu62, Tea80, AMP84, TC88].

Matching in global feature methods is usually very simple; most of the effort goes into computing the features. The preprocessing stage creates a multiview feature representation for each model by calculating the feature vector for each model for a large number of views, usually approximately uniformly spaced over the viewing sphere. Most methods do this by graphically rendering the silhouette of 3-d CAD models of the objects, although some have simply taken images of a physical model over the viewing sphere [DBM77]. At recognition time, an image is processed by segmenting the object region and calculating the feature vector from it. Then, the feature vectors stored in the

multiview models are compared to the image feature vector using some distance measure on the feature space. The model and view with the most similar global feature vector to the image feature vector is taken to be the correct identification of the object. Typically, no further verification is done.

A good example of a global feature method is that described by Dudani *et al* in [DBM77]. The features used by this method are *moment invariants*. There are several flavors of moments and invariants. Dudani *et al* use the standard central moments and the invariants found by Hu [Hu62] using the theory of algebraic invariants. A moment is defined as:

$$m_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^p y^q \rho(x,y)\,dx\,dy, \qquad p,q = 0,1,2,\cdots, \tag{3.2}$$

where $\rho(x,y)$ is an image function. In the method at hand, if S is the set of silhouette points in the image, then $\rho = 1$ if $(x,y) \in S$, and $\rho = 0$ otherwise. The central moments are defined by

$$\mu_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} (x-\overline{x})^p (y-\overline{y})^q \rho(x,y)\,dx\,dy, \qquad p,q = 0,1,2,\cdots \tag{3.3}$$

where

$$\overline{x} = m_{10}/m_{00}, \overline{y} = m_{01}/m_{00}.$$

Central moments are invariant to translations of the silhouette. Using the theory of algebraic invariants, it is possible to combine the central moments so that they are invariant to image plane rotation and scaling as well. For example, the invariants of order two, i.e., $p+q=2$, are:

$$\mu_{02} + \mu_{20},$$

$$(\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2.$$

Dudani *et al* used the seven lowest order invariants. Higher order invariants have been shown to be sensitive to noise [Wie83, AMP84, AMP85, TC88]. The feature vector was fourteen elements long: seven of the invariants were computed with S as the entire

silhouette region (while the other seven were computed using S as only the points on the silhouette boundary.

A multiview model was constructed by taking images of the models to be recognized at 5° increments of the Euler angles parameterizing the viewing directions, and, for each view, calculating and storing the resulting fourteen element feature vector.

Recognition consisted of taking the binary silhouette, calculating its 14 moment invariants, and searching the multiview models for the feature vectors that are most similar using the distance-weighted k-nearest neighbor rule [Dud76]. The viewing direction that yielded the most similar feature is reported as the system's recognition result. This system does not actually make a decision about whether the object is present or not, and so arguably, does not perform true recognition.

Other global, moment-based methods include those described in [RT89, TR87, BF86, Ree81, RPT85]. There are a number of methods based on normalized Fourier descriptors as well [WM80, WW80, WMF81, Kuh84, SD71].

3.2.5 Optimization-Based Methods

Optimization based approaches are superficially similar to the AEFMA module described in this thesis. These methods usually generate some kind of global disparity measure based on the shape-disparity between curves or regions in the image and the rendering of curves or silhouettes derived from a 3-d model. The disparity measure is a function of the viewing parameters. Since the viewing parameters form a continuous space, and the disparity measures are themselves made to be smooth functions of the viewing parameters, continuous optimization procedures may then be used to search the space of scene instances for the minimum of the disparity measure. If the disparity resulting from applying the optimization procedure is small enough, an object is recognized.

While the approach is, in principle, sound, there are a number of difficulties that plague these methods. The most serious is the problem of local minima. Any continuous

optimization procedure uses local knowledge of the similarity function to drive its search. Unfortunately, local information yields no information enabling a local minimum to be distinguished from the global minimum, thus there is no way for an optimization procedure to distinguish a local mimima from a global mimima. Therefore, these methods often become trapped in local minima of the disparity measure.

Another problem is the disparity function itself. The nature of the disparity measure impacts the number and severity of the local minima, as well as whether the method can find partial instances of objects. The problem of local mimima is often dealt with by starting the optimization at many different points in scene-instance space, choosing the best result, and hoping that the best result is the global minimum. This is both time-consuming and unreliable. Recognition of partial objects is not possible with the reported methods.

While optimization-based methods have difficulties, they have the advantage that any features, not just object-attached features, can be used. If their other problems could be solved, they would be able recognize wider classes of objects than other methods. The AEFMA approach developed in this thesis goes much of the way toward solving these problems, opening the door to this goal.

A recent optimization-based method possessing some parallels with Cyclops is described in by Stark *et al* [SEB88]. Models in this method are polyhedra and an associated *aspect graph*. As mentioned previously, an aspect graph is a set of topological equivalence classes over the sphere of viewing directions. In this case, the edges of the polyhedral models are rendered under perspective, with hidden lines removed, and the resulting views are grouped by the equivalence of the topology of the resulting set of 2-d line segments. Each such equivalence class is called a "cell". Associated with each cell is a set of viewing parameters that generate a "prototype" instance of the object. The essence of the method is to use these prototype model instances as starting points for optimization. The authors argue that the aspect graph provides a partition of viewing

parameter space that will be likely to have separate local minima. Thus, the optimization is started from *every* prototype model instance, and it is constrained to remain in the cell during the optimization. The best result is then selected as the recognized object.

The idea of attempting to systematically isolate local minima is a good one, and is somewhat similar to the approach of Cyclops. However, Cyclops uses a feature indexed associative memory to retrieve *only* those views that have matching features rather than attempting to start from every possible stored view, or aspect, as this method does.

While there are some parallels to Cyclops the optimization portion of the algorithm is rather different. The system processes an image to extract a line drawing of the image, and, treating it as a graph with vertices as nodes and segments as arcs, selects a unique subset of the set of all possible circuits in the graph. These circuits are the "elementry" circuits of the graph, and correspond to the faces of the imaged polyhedron. Next, a Fourier descriptor representation of each such elementry circuit is computed. The figure of merit for the match is computed by projecting the model using the current set of viewing parameters (as obtained from the optimization algorithm), and Fourier descriptor features are computed as for the image features. The algorithm examines all possible pairs of image circuit Fourier descriptor features with those derived from the model to determine the "best" match. Using this match, a "figure of merit" is reported (not described in the paper) that is used as the cost function. The cost function is optimized using the method of damped least squares.

Note that this method requires the entire object to be visible. In addition, the method works only for simple polyhedra, as complex polyhedra have intractably large aspect graphs.

There are a number of earlier examples of optimization-based methods. The method described in [WS82] uses the Levenberg-Marquardt method to optimize the squared distance between the Fourier descriptor representations of the image edge contours and the projection of a space curve. The approach described in [HW75] uses gradient descent

to optimize a matching metric consisting of the sum of distances between corresponding points on the image edge contours and the silhouette outline. The system described in [MGA88] is interesting because it describes connectionist matching network, and, further, includes both a subpart hierarchy for models and a class hierarchy, expressed in the form of **isa** and **ina** links in the model network. Possible instantiations of model entities are connected to all possible matching model parts and subparts by matching nodes. The strength of the matching nodes are adusted to maximize the consistency of the match. The consistency of a match is computed using "consistency rectangles" which, essentially, describe the similarity of binary relations in the image to binary relations in the model.

Local minima are reported to be problematic in all of the methods described above, resulting in limited success. In all cases, continuous optimization methods were used. Discrete optimization methods, such as simulated annealing [Rut89] and dynamic programming [ATW88] could be used, but they are more inefficient than continuous methods. A better solution is to use additional information to provide good guesses as to the location of the global minima. This idea is at the heart of Cyclops, and opens the way to the recognition of occluded objects using general, not just object-attached features.

CHAPTER 4

HYPOTHESIS GENERATION AND RELATED TOPICS

As described briefly in Chapter 2, Cyclops employs a variation of the classical hypothesize and verify paradigm. This chapter focuses on issues involved in *hypothesis generation.* To achieve reasonable performance, it is necessary to consider all the factors that influence it. In particular, both the choice of features and the representation of the models have critical impact on the hypothesis generator. The processes and data structures of the Cyclops framework that this chapter is concerned with are shown in Fig. 4.1. Efficiently generating good hypotheses is critical to the success of the overall algorithm. The design of the feature grouping and feature detection processes as well as the multiview model associative database and the feature-based image representation are driven by the requirements encountered in the design of the hypothesis generator, and, therefore, are discussed in this chapter.

The approach to hypothesis generation employed in Cyclops differs in a very important respect from approaches employed by other HVP methods, such as those described in Chapter 3 Section 3.2.2.1. Without exception, such methods employ the simplifying assumption that the features are object-attached. Since Cyclops does not employ object-attached features, Cyclops' hypothesis generator is necessarily more complex. However, unlike existing methods, Cyclops' hypothesis generator is able to handle objects of any shape, not just the small subset of shapes that reliably generate object-attached features.

The primary objective of Cyclops' hypothesis generator is to take an image feature, and return a set of model instances that, when rendered, generate a feature that is sim-

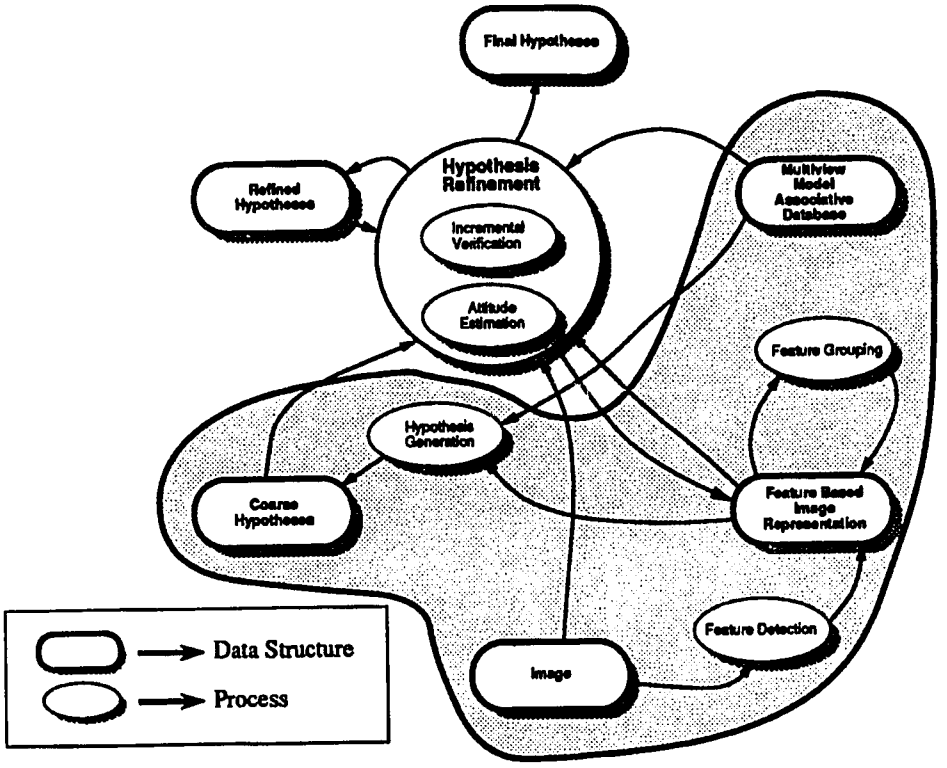


Figure 4.1. The parts of Cyclops that are treated in Chapter 4

ilar to the observed feature, to within tolerances determined by noise, quantization and other factors. Thus, the hypothesis generator is actually performing *backprojection*, as described in Chapter 3, Section 3.2.3.1. The set of model instances returned by the hypothesis generator should satisfy the following criteria:

1. the probability that a correct solution is excluded from the set is very small, and
2. the set should be as small as possible.

The requirement that the set does not exclude a correct solution ensures that the hypothesis generator will not cause Cyclops to miss an object. Making the set as small as possible narrows the search, thus improving the efficiency of the remainder of the algorithm.

Of all of the modules of Cyclops, the hypothesis generator performs the most drastic reduction in the size of the set of feasible scene instances. Prior to the invocation of the hypothesis generator, all model instances must be considered equal candidates for recognition. After invocation, the hypothesis generator will have returned a small

number of model instances that could explain the observed features. Thus, the hypothesis generator must consider the entire space of model instances as it searches for those that contain matching features. We demonstrated the immensity of the space of scene instances and model instances in Chapter 1. The hypothesis generator must be carefully designed in order to not require an intractably large amount of computation.

Previous approaches to 3-d recognition, where object-attached features are assumed, hypothesis generation is a relatively trivial problem. Typically, as described in Chapter 3, the approach is to put primitive image features into a hypothetical correspondence with 3-d model features until enough constraints exist to nearly uniquely determine the viewing parameters of a hypothesized model. Projecting the model using these viewing parameters yields 2-d features that have similar attributes to the image features. The resulting model instances are then used to predict more completely what appears in the image. Accurate predictions are considered as evidence toward verification of the hypothesis, while inaccurate predictions are considered as negative or neutral evidence. The most difficult aspects of such approaches include computation of the viewing parameters and, to a lesser extent, the choice features to use. If enough model-feature to image-feature correspondences are made, then the problem becomes overconstrained, necessitating the iterative solution of a non-linear least squares problem, as, for example, in Lowe [Low85, Low87a, Low87b]. If the problem is neither underconstrained nor overconstrained, then analytic solutions have been found. For example, [FB81] describes a solution for perspective, [Hut88, Chi89] describe solutions for weak perspective, as well as numerous other examples for these and other cases. Whether or not an iterative or an analytic solution is used, the problem is well-conditioned and obtaining a solution is simple.

The hypothesis generator in Cyclops cannot exploit the simplifications resulting from the object-attached feature assumption, since, in the general case, there is not a simple correspondence between surface patches and features that holds over a range of views.

In fact, as shown in Chapter 1, the attributes of general features cannot be predicted by simply projecting *a priori* known patches of model surfaces, i.e., the patches that reliably generate object-attached features. Precisely because of the fact that they have employed object-attached features, previous 3-d recognition algorithms employing the hypothesize-and-verify paradigm have taken this approach exclusively. Hypothesis generation using general features, as in the Cyclops framework, becomes more complex. However, many of the same issues arise, especially in the interplay between the features and the hypothesis generation process.

4.1 Avoiding Object-Attached Feature Assumption: The Impact on the Hypothesis Generation Process

We have shown that object recognition systems that do not rely on the object attached feature assumption can handle larger classes of objects than systems that do employ the assumption. To our knowledge, Cyclops is the first system designed to recognize 3-d objects using spatially local, non-object-attached features. This is one of Cyclops' primary innovations. We have also alluded to the fact that hypothesis generation is more complex using general features than using object-attached features. The essential reason for this is that, in order to predict the location of general features, the object must first be graphically *rendered* using the viewing parameters. Only then can the features be predicted by detecting them in the graphical rendering. As we will see, this implies that, in order to *efficiently* generate hypotheses, some form of *multiview model*, as introduced in Chapter 2, must be used to represent the object. We now show why rendering of a scene is necessary to predict general features, and why this leads to adopting a multiview representation of the models.

In the context of this work, rendering is the process of predicting the appearance of a scene or an object by the methods of computer graphics. Generally, rendering employs time consuming hidden line and surface removal algorithms to determine which portions

of a 3-d object are visible given a set of viewing parameters. To accurately predict the sensed pixel values, further information, such as position, intensity, and spectral characteristics of light sources as well as the reflectance properties of all surfaces are necessary. Fortunately, it is seldom necessary to do this since we usually desire abstractions of the image such as edge contours. Such abstractions can usually be predicted reasonably well from purely geometric information. Nevertheless, some form of rendering is, in general, required even to predict edge contours, and, therefore, to predict the attributes of features derived from edge contours such as high curvature points, inflection points, and linear segments.

Contrast this to the prediction of object-attached features, where the location and attributes such features can be calculated by simply projecting a representation of a particular patch of the 3-d surface of the model; the particular patch of surface that generates the feature is assumed to be independent of viewpoint, obviating the need to perform complex rendering. For example, to predict the pose in the image of the corner of a cube, and the edges incident on it, simply projecting the 3-d vertex of the cube and the three line segments comprising it is enough. Fig. 4.2 depicts this case. In comparison, Fig. 4.3 illustrates how the portion of the surface that generates a feature may change drastically when general features are used. There, the portion of the surface of the object that projects to the high curvature point can be seen to move about as the viewpoint changes.

Now consider efficiently generating hypotheses using non-object-attached features. Recall that the goal of a hypothesis generator is to select model instances possessing features whose attributes nearly match the attributes of an image feature. Given such an image feature, if we were to use *only* the 3-d representation of the object, it would be necessary to render the object at many points in the set of all possible viewing parameters in search of the model instances that yield features with similar attributes. This would be very inefficient. The only alternative is to render and compute features *in advance*, at

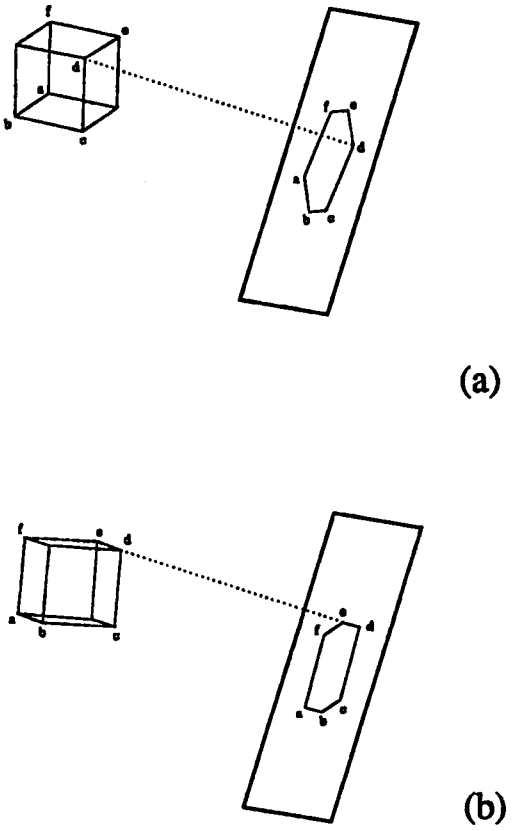
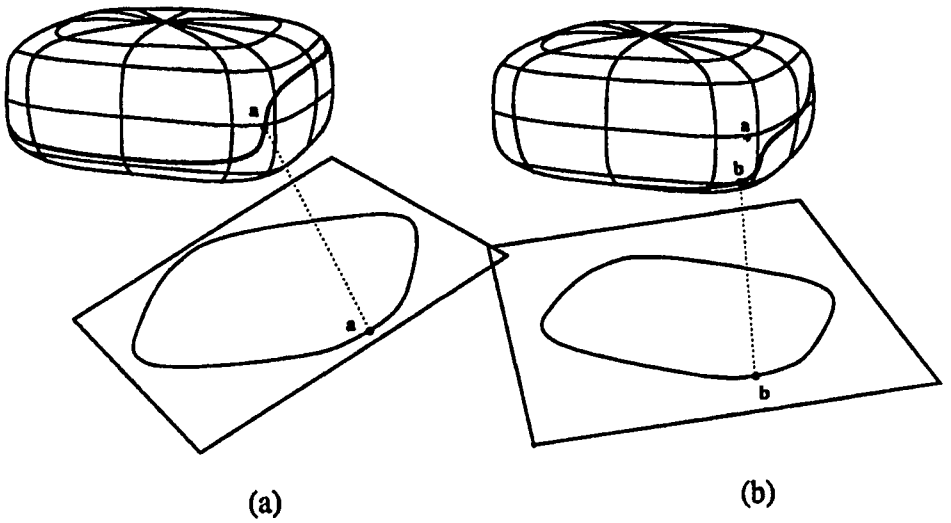


Figure 4.2. Shown in (a) and (b) are the silhouetted boundaries that result from the projections of a cube onto a viewer's image plane, where the viewing directions in (a) and (b) differ substantially. The vertices of the cube that project to the vertices of the silhouette polygon are indicated by the same letters. Note that the vertices of the cube, vertex d for example, project to vertices of the silhouette polygon even though the viewpoint is markedly different. This is true for a wide range of viewpoints where vectex d of the cube is visible. Thus, since we know *a priori* that vertex d on the cube will always project to vertex d of the polygon, independent of viewpoint, all that is required to predict the appearance of vertex d of the polygon is to project a representation of the vertex d of the cube. This behavior characterizes object-attached features.

many points in the space of viewing parameters, and place them in a database for access during recognition. This is the key to Cyclops' approach to hypothesis generation.



(a) (b)

Figure 4.3. In a manner analogous to Fig 4.2, (a) and (b) show the silhouette boundary resulting from the projection of a smooth object onto the viewer's plane from two viewpoints. In (a), a high curvature point, marked **a**, is shown along with the point on the surface of the object that projected to **a** in the silhouette boundary, also marked **a**. The same situation, although from a different viewpoint, is shown in (b). In this case, the high curvature point and its inverse image on the surface of the object are labeled **b**. The location of **a** is also shown on the surface of object. Clearly, the locations of **a** and **b** are not the same, showing that the points on the surface of a smooth object that produce high curvature points are dependent on the viewing parameters. Thus, high curvature points on a smooth object are *not* object attached. A similar demonstration is possible for inflection points and line segments.

If the attributes of the features used to represent the model instance are approximately *invariant* to changes in one or more of the viewing parameters, then the number of model instances that need to be stored in the database can be reduced drastically. This is because large regions of viewing parameter space can be represented by a single model instance. Thus, a small number of model instances may suffice to adequately represent the entire space of model instances. Under weak perspective, it is possible to construct features that are invariant to all image plane transformations that preserve shape. Further, by

using edge-based features, sensitivity to lighting parameters can be reduced. Thus, it is largely possible to parameterize model instances only by using rotations out of the image plane. This is a 2-d subspace of the viewing parameter space. Essentially, this subspace of viewing parameter space parameterizes the *views* of the object. This is why we have chosen to call the feature-indexed database of model instances the *multiview feature representation* of the model.

The approach outlined in the previous paragraph can improve the efficiency of the hypothesis generation process in two immediate ways. First, and most obviously, since the features are precomputed, the computation required to render the object and then detect the features is avoided during recognition. Second, and less obviously, since we have precomputed the features, we have the freedom to *index* the database of model instances to permit efficient associative access based on the attributes of the features. When the set of model instances is viewed as a database to be accessed based on the attributes of the features, the idea of indexing is obvious. To our knowledge, this freedom has not been previously exploited until the work leading up to this thesis [GTM87, GTM89]. Typically, when precomputation is used, a simple linear search through the database of features is performed to find the model instances containing features similar to an image feature. Cyclops exploits the freedom to index features, using a data structure that is ideally suited for this purpose.

4.2 Indexing Model Views by Feature Attributes

Given a feature that has been detected in the image, we wish to find model instances that contain features with similar attributes, as efficiently as possible. This problem is at the heart of Cyclops' hypothesis generation process as well as the design of the multiview associative database. In order to achieve our goals, we cast the hypothesis generation problem as a problem in *range searching*, a problem that has been studied extensively. In what follows, we first examine how the hypothesis generation process may be viewed

as a range search, and then show how to implement hypothesis generation efficiently by implementing the multiview associative database using a range searching data structure called a k-d tree.

4.2.1 Hypothesis Generation Viewed as Neighborhood Searches in Feature-Attribute Space

Given some type of features that are invariant to variations in in-plane rotation, translation, and scale, the feature attributes of a particular image feature will never exactly equal the attributes of a precomputed, model feature, even when the features correctly match, except in very unlikely circumstances. Noise in the imaging process is one source of distortion. However, noise may be of minor importance compared to the distortion introduced by quantizing the viewing sphere. The multiview models employed in Cyclops represent regions of the viewing sphere by features detected in *representative views*. The set of representative views is discrete, and the actual view of an object recorded by the image will generally fall between the representative views. Since, in general, the attributes of features are functions of viewpoint, there will be differences in the attributes of the the image feature and the corresponding model feature, even in the nearest views to the actual view. Essentially, the *forward projection* of each representative view region of the viewing sphere maps to a region of feature-attribute space. These regions are not uniform, as the attributes of a feature may change quickly over one region of the viewing sphere, and less quickly elsewhere. It is important to note that, if the views are spaced arbitrarily closely over the viewing sphere, the possible distance in feature-attribute space between correctly matching image features and precomputed model features can be made arbitrarily small. This is because the feature attributes are assumed to be continuous functions of the viewpoint. This assumption will be discussed more fully later in the chapter.

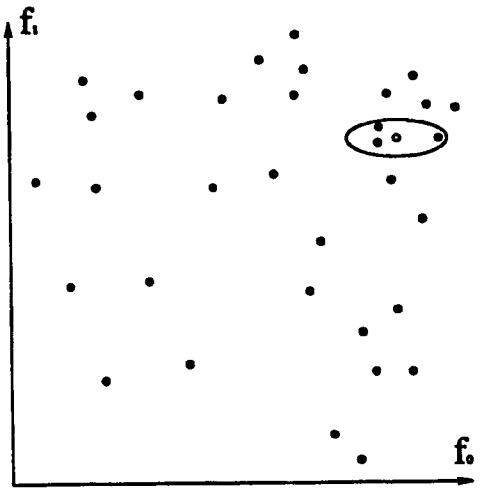


Figure 4.4. An example of a neighborhood query.

Regardless of the source of the distortion, correctly matching image features and model features possess attributes that are nearly equal, though never exactly identical, in feature-attribute space. Thus, given an observed image feature, the database of model instances must allow efficient retrieval of those model instances that have a feature falling in a *neighborhood* of the image feature in feature-attribute space. This concept is illustrated in Fig. 4.4. The optimal size of the neighborhood depends on three interrelated factors: the amount of noise, the spacing of the views over the viewing sphere, and the desired probability that an object present in an image not be missed. We will discuss the interplay of these factors in greater detail later. For now, note that if the neighborhood is made too small, the probability that the correct model instance will be retrieved will become unacceptably small since distortion of the image feature will be likely to move the center of the query neighborhood far enough that the correctly matching model feature will fall outside the neighborhood, forcing the probability of a miss to become unacceptably large. This situation is shown in Fig. 4.5(a). On the other hand, if the neighborhood is too large, then the true model instance will be retrieved, however, many false hypothesis will be generated as well, leading to both wasted verification effort and to the possibility of false positives. This case is shown in Fig. 4.5(b).

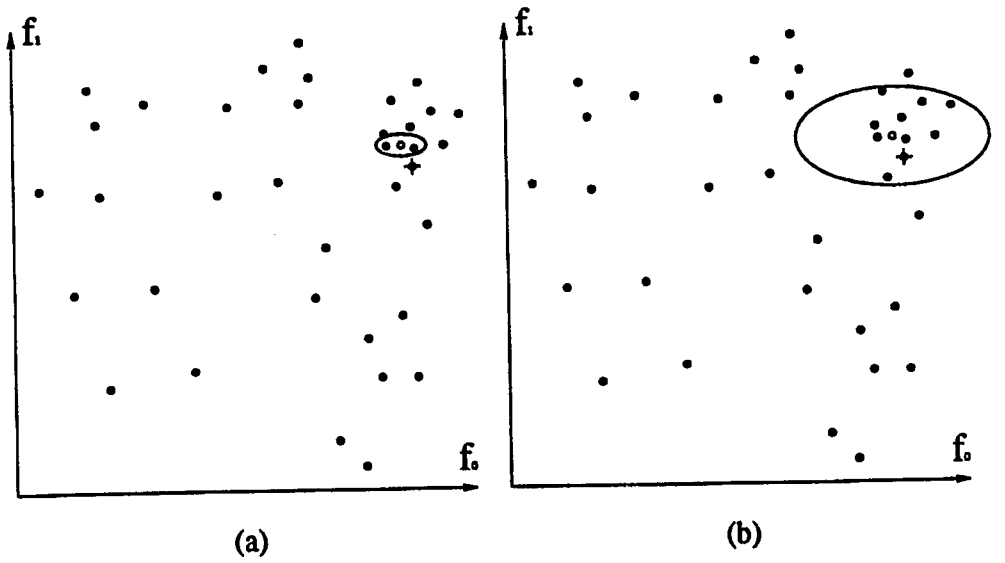


Figure 4.5. A neighborhood query that is too small is shown in (a). The image feature is the hollow dot, and the correctly matching model feature is a dot with a cross. Other, incorrectly matching model features are shown as solid dots. The query neighborhood is too small because the it does not include the correctly matching feature. In (b) the neighborhood is large enough to include the correctly matching feature, however, it is larger than necessary, resulting in many other possible matches (solid dots) falling in the query neighborhood.

4.2.2 Casting Neighborhood Searching as Range Searching

A *range search* is a type of query on a database of vectors possessing ordered components. Specifically, given a set V of k-dimensional vectors, then a range search of V asks for R, the subset of V such that each member $\mathbf{r} \in R$, satisfies $r_i \in [l_i, h_i], i = 1 \ldots k$, where the r_i are the components of $\mathbf{r}$, and the intervals $[l_i, r_i]$ are the *ranges* of the range search. That is, a range search returns all the vectors in a set having components that fall simultaneously into ranges.

Viewed geometrically, a range search in a multidimensional vector space queries the database for all vectors falling within a multidimensional rectangle. Fig. 4.6 shows the

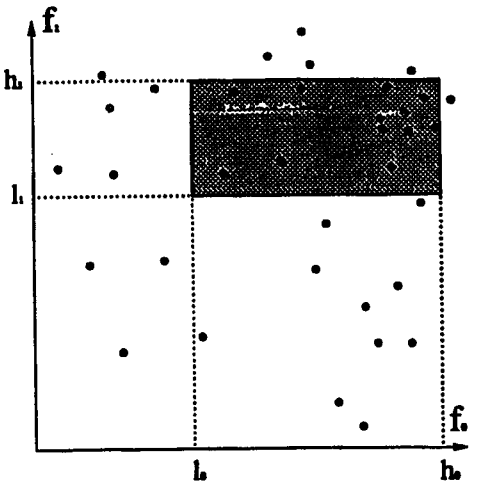


Figure 4.6. Geometric view of range queries.

geometric interpretation of a range search on a 2-d vector space. The extension to higher dimensions is simple.

Having defined *range searching* we now turn to the definition of a *neighborhood* and *neighborhood searching*. A delta neighborhood of a vector $\mathbf{p}$, denoted as $N(\mathbf{p}, \delta)$ is a set of points $\{\mathbf{x} | d(\mathbf{x}, \mathbf{p}) < \delta\}$, where $d(\mathbf{a}, \mathbf{b})$ is a distance measure between two vectors $\mathbf{a}$ and $\mathbf{b}$. A common distance measure is derived from the weighted l-norm:

$$d_l(\mathbf{a}, \mathbf{b}) = \left[\sum_{i=1}^{N} \left(\frac{|a_i - b_i|}{\alpha_i} \right)^l \right]^{\frac{1}{l}} \tag{4.1}$$

where the $\alpha_i \in (0, 1]$ are weights. As $l \to \infty$, this becomes

$$d_\infty(\mathbf{a}, \mathbf{b}) = \max_{i=1}^{N} \frac{|a_i - b_i|}{\alpha_i}. \tag{4.2}$$

The weighted l-metric is important because the following section shows that neighborhood queries using the weighted l-norm can be implemented very efficiently. Let $N_l(\mathbf{p}, \delta)$ be the delta neighborhood of vector $\mathbf{p}$ under the d_l metric. We first show that a neighborhood query when $l = \infty$ reduces a range query. We then show that $N_\infty(\mathbf{p}, \delta) \supseteq N_l(\mathbf{p}, \delta)$, where $l \in (1, \infty)$. Thus, a $N_l(\mathbf{p}, \delta)$ query can be accomplished by first doing a N_∞ search followed by a filter to remove any vectors that do not fall in $N_l(\mathbf{p}, \delta)$. This is important since the following section shows that range queries can be accomplished very

efficiently, and an $N_\infty(\mathbf{p}, \delta)$ reduces to a range query, implying that an $N_l(\mathbf{p}, \delta)$ query can also be performed efficiently.

We turn first to showing that a delta neighborhood of a vector $\mathbf{p}$ under the d_∞ metric reduces to a *range* about the components of $\mathbf{p}$. The δ neighborhood induced by d_∞ about $\mathbf{p}$ is the set

$$N_\infty(\mathbf{p}, \delta) = \left\{ \mathbf{x} \mid \max_i^N \frac{|x_i - p_i|}{\alpha_i} < \delta \right\}. \tag{4.3}$$

Let k be the index yielding the maximum value of $|x_i - p_i|/\alpha_i$ for some vector $\mathbf{x} \in N_\infty(\mathbf{p}, \delta)$. By definition,

$$\frac{|x_k - p_k|}{\alpha_k} < \delta.$$

Thus,

$$x_k \in [p_k - \delta_k, p_k + \delta_k],$$

where $\delta_k = \delta\alpha_k$. Since

$$\frac{|x_i - p_i|}{\alpha_i} \leq \frac{|x_k - p_k|}{\alpha_k} < \delta,$$

the same argument as above yields

$$x_i \in [p_i - \delta_i, p_i + \delta_i], i \ldots N. \tag{4.4}$$

Equation (4.4) simply states that if $\mathbf{x} \in N(\mathbf{p}, \delta)$, then the components of $\mathbf{x}$ fall in a multidimensional range, where the components of $\mathbf{p}$ are at the center of each range.

We now show that $N_\infty(\mathbf{p}, \delta)$ contains $N_l(\mathbf{p}, \delta)$ for all $l \geq 1$. First, $d_l(\mathbf{a}, \mathbf{b}) \geq d_\infty(\mathbf{a}, \mathbf{b})$ for all $l \geq 1$ since

$$d_l(\mathbf{a}, \mathbf{b}) = \left[\sum_{i=1}^N \left(\frac{|a_i - b_i|}{\alpha_i} \right)^l \right]^{\frac{1}{l}} \geq \frac{|a_k - b_k|}{\alpha_k} = \max_{i=1}^N \frac{|a_i - b_i|}{\alpha_i} = d_\infty(\mathbf{a}, \mathbf{b}),$$

where, as above, the k^{th} term is the largest out of

$$\frac{|a_i - b_i|}{\alpha_i}, i \ldots N.$$

Now, let $\mathbf{x} \in N_l(\mathbf{p}, \delta)$. Then, by definition, $d_l(\mathbf{p}, \mathbf{x}) < \delta$. But we just showed that $d_l(\mathbf{a}, \mathbf{b}) \geq d_\infty(\mathbf{a}, \mathbf{b})$ for all $l \geq 1$. Thus, $d_\infty(\mathbf{p}, \mathbf{x}) \leq d_l(\mathbf{p}, \mathbf{x}) < \delta$, $l \geq 1$, implying that $\mathbf{x} \in N_\infty(\mathbf{p}, \delta)$ as well. Thus, $N_l(\mathbf{p}, \delta) \subseteq N_\infty(\mathbf{p}, \delta)$, $l \geq 1$.

4.2.3 Efficient Implementation of Neighborhood Searches

We have seen that one of the key problems in Cyclops is finding which set of model instances, out of a larger set that is stored in a database, possess features similar to a feature that has been detected in the image. We have also seen that a model-generated feature is similar to an image feature if the model-generated feature is contained in a neighborhood of the image feature, in feature-attribute space. Thus the problem of generating hypotheses reduces to the problem of searching the set of model-generated feature vectors for all vectors in the set that fall within a neighborhood of the image feature. Due both to the potential size of the database of model instances, and due to the large number of image features, each possibly requiring a neighborhood query, it is critical that the query operation be implemented in an efficient manner as possible.

The key to efficient implementation of neighborhood queries is efficient implementation of range searching. As mentioned previously, a *range* is a multidimensional interval. That is, a range $R \subset \Re^n$ is defined as

$$\{\mathbf{x} \mid x_i \in [l_i, h_i]\},\ i \ldots k,$$

where the l_i and the h_i define the lower and upper limits respectively of each component of the vector x. Given a finite set of k-d vectors, S, a range query returns the elements of S that are contained in R.

Data structures for implementing range searches have been well studied [Ben75, BF79, BM79]. Table 4.1 gives a summary of the known algorithms. The important factors characterizing such algorithms include

- search complexity,
- build complexity,
- storage complexity, and
- implementation complexity.

The paramount consideration in Cyclops' hypothesis generation process is efficiency. Thus, search complexity is the most important factor. The building of the data structure is done offline, and, therefore, is not important so long as it can be accomplished in reasonable time. Similarly, none of the range searching techniques has high enough storage complexity to warrant much influence on the decision. Implementation complexity is a factor, though considerably less important than the search complexity.

For use in Cyclops we have chosen to use the k-d tree data structure. There are several reasons for this. First, as can be seen from the table, k-d trees have the best query complexity, $O(\log n + f)$, where n is the size of the database, and f is the size of the returned set. This is average case complexity, and this is appropriate for Cyclops since the probability that the linear worst case complexity would be realized is infinitesimal. In addition, the algorithms possessing the best worst-case query complexity, range trees and overlapping k-ranges, also have the highest implementation complexity. Considering the extremely small probability of the worst-case situation, the additional cost in implementation complexity is not deemed worthwhile. As a final point in their favor, k-d trees also possess excellent build and storage complexity.

As indicated by its name, a k-d tree is a hierarchical data structure for indexing multidimensional vectors for rapid retrieval. If the vector space is viewed as a Euclidean space, a k-d tree functions somewhat like a quad-tree in two dimensions, or an octree in three dimensions, in that it recursively subdivides the vector space. Unlike quadtrees or octrees, a k-d tree divides the vector space into regions such that there are an approximately equal number of vectors in each region. The regions are formed by "splitting" the space recursively until there are a small number of vectors in each region. This process is illustrated in Fig. 4.7. There, each line corresponds to a node of the tree. Lines at deeper levels of the tree are drawn thinner than those near the root. Fig. 4.8 shows geometrically how a query of a k-d tree is performed.

Building a k-d tree is a simple matter, and, in all but the most degenerate circum-

Data Structure	$B(n,k)$	$S(n,k)$	$Q(n,k)$	I
Sequential Scan	$O(n)$	$O(n)$	$O(n)$	low
Projection	$O(n \log n)$	$O(n)$	$O(n^{1-1/k} + f)^*$	low
k-d trees	$O(n \log n)$	$O(n)$	$O(\log n + f)^*$	med
Non-Overlapping k-ranges	$O(n \log n)$	$O(n)$	$O(n^\epsilon + f)$	med
Range Trees	$O(n \log^{k-1} n)$	$O(n \log^{k-1} n)$	$O(\log^k n + f)$	high
Overlapping k-ranges	$O(n^{1+\epsilon})$	$O(n^{1+\epsilon})$	$O(\log n + f)$	high

Table 4.1. Algorithms for range searching. In the table, $B(n,k)$ denotes the complexity of building the data structure; $S(n,k)$ denotes the space complexity, and $Q(n,k)$ denotes the complexity of the range query, where n is the number of vectors in the database, and k is the dimension of the vectors, and F is the size of the returned set of vectors, i.e., the number vectors contained in the range. An asterix denotes average-case complexity. Others are worst-case.

stances, we can insure that the tree will be balanced. The tree is constructed by choosing a component of the vector space along which to divide the vectors. Let this component be the n^{th} component. Let the vector having its n^{th} component as the median of all the vectors in the set be $\mathbf{m}^n$. The set is then divided into two sets of equal or nearly equal size, one set whose members have their n^{th} components less than or equal to the n^{th} component of $\mathbf{m}^n$. This set will be the *left child* of the current node after recursively building a subtree out of the set. The other set, whose members have their n^{th} components greater than the n^{th} component of $\mathbf{m}^n$ will, in a similar manner, eventually become the *right child* of the current node. The current node holds the value of n, the value of the n^{th} component of $\mathbf{m}^n$, and the pointers to the left and right children of the node. The left and right subtrees of the current node are constructed recursively in exactly the same manner. The recursion stops when the sets become smaller than a preset limit, typically

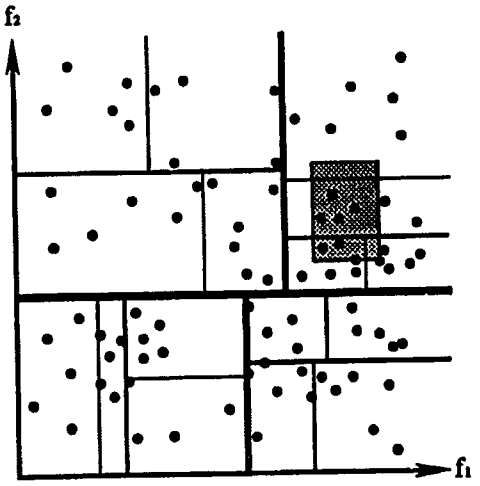


Figure 4.7. A hypothetical 2-d feature space partitioned by a k-d tree. Dots represent the 2-d vectors used to index the tree. Each line represents a node in the tree. Thicker lines represent nodes closer to the root. Range queries represent arbitrary rectangles, such as the one shown in gray.

three to six elements. Fig. 4.9 shows an example of a 3-d k-d tree, and shows the path of the nodes explored for an example query.

We have now described how to perform range searches very efficiently. This, as shown earlier, is equivalent to performing a $N_\infty(\mathbf{p}, \delta)$ neighborhood search. However, we have not yet shown how k-d trees can be used to perform general $N_l(\mathbf{p}, \delta)$ neighborhood searches. To do this, we use the result of the previous section that $N_l(\mathbf{p}, \delta) \subseteq N_\infty(\mathbf{p}, \delta)$. Thus, to perform an $N_l(\mathbf{p}, \delta)$ neighborhood query, we first do a $N_\infty(\mathbf{p}, \delta)$ neighborhood query using a k-d tree. Finding $N_\infty(\mathbf{p}, \delta)$ from $N_l(\mathbf{p}, \delta)$ is a simple matter given the weights, i.e., the α_i in Eq. 4.3. Then, by virtue of the fact that $N_l(\mathbf{p}, \delta) \subseteq N_\infty(\mathbf{p}, \delta)$, we can simply sequentially scan this set to remove any of the vectors in $N_\infty(\mathbf{p}, \delta)$ that are not in $N_l(\mathbf{p}, \delta)$.

The scanning operation described in the previous paragraph is linear in the size of the returned set. Thus, the question arises as to whether an $N_l(\mathbf{p}, \delta)$ neighborhood query has the same complexity as an $N_\infty(\mathbf{p}, \delta)$ neighborhood query. The answer is yes, and can be easily demonstrated. From Table 4.1 we see that the average case complexity

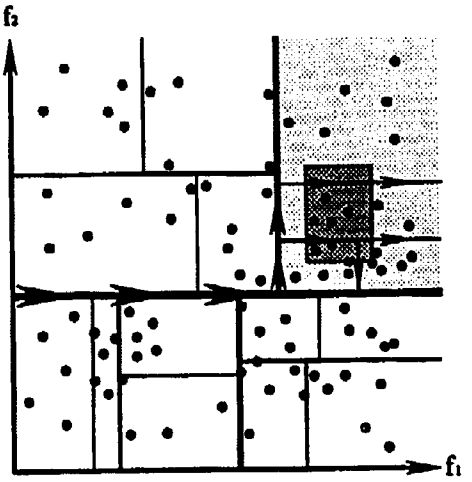


Figure 4.8. Querying a k-d tree. The task is to find all vectors that lie within the gray rectangle, which is an example of a 2-d range. Starting at the root, all branches are are explored that bound a region that contains any part of the rectangle. The branches are traced until the all of the leaf regions , shown in light gray, that contain any part of the rectangle, shown in dark gray, are identified. The vectors contained in these regions are then scanned to remove those that do not lie within the rectangle. This process can be accomplished in $O(\log n)$ time.

of a $N_\infty(\mathbf{p}, \delta)$ neighborhood query is $O(\log n + f)$, where n is the number of vectors in the database, and f is the average size of the returned set. The complexity of the scanning operation is $O(f)$. The total complexity of a $N_l(\mathbf{p}, \delta)$ neighborhood query is $O(\log n + f) + O(f) = O(\log n + f)$. Thus, a $N_l(\mathbf{p}, \delta)$ neighborhood query has the same complexity as a $N_\infty(\mathbf{p}, \delta)$ neighborhood query.

Cyclops uses a k-d tree to implement a database of model instances and index them on the attributes of the features predicted to appear in each. Thus, given a feature that has been detected in the image, Cyclops uses the k-d tree to efficiently perform a weighted $N_2(\mathbf{p}, \delta)$ neighborhood query to obtain the model instances having features that possess similar attributes, and therefore have similar appearance. However, efficient retrieval is useless if the query returns a large number of model instances. This may occur in two ways: either the features have not been chosen properly, or there are many

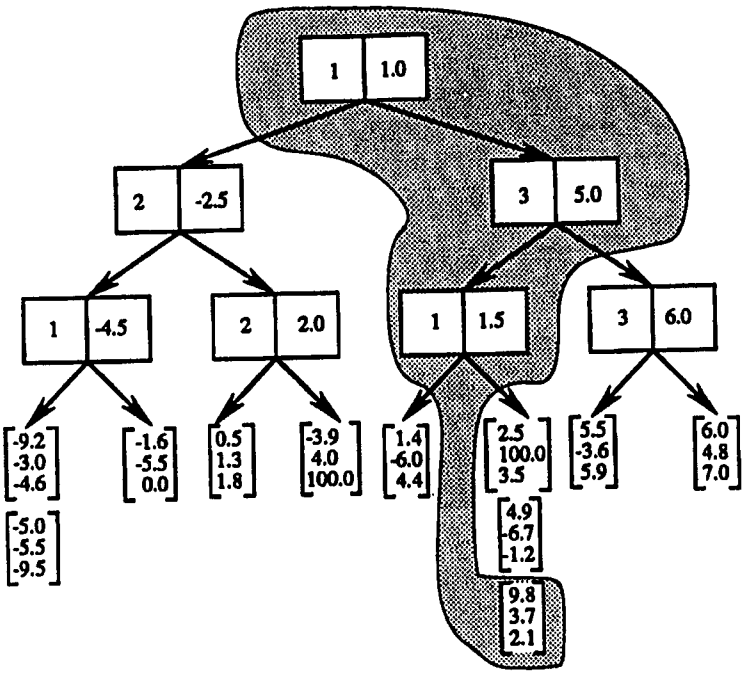


Figure 4.9. Above is a simple k-d tree for rapid retrieval of 3-d vectors. The enclosed area shows the sequence of nodes traversed to retrieve the one vector in the tree that is in the neighborhood $(10.0, 4.0, 2.0)^T + (\pm.5, \pm.5, \pm.5)^T$.

redundant model instances stored in the database. The next section discusses the selecting of features in Cyclops so as to avoid these problems, and enhance the performance of the system. The section following it discusses the problem of redundant model instances in the context of multiview models.

4.3 Features for Hypothesis Generation in Cyclops

In much of the previous work in object recognition, a great deal of effort has been spent on the problem of feature selection. In particular, effort is often spent to detect features possessing special properties that researchers believe will simplify other parts of the object recognition system. One example of this are the various global features, such as Fourier descriptors and moment invariants, discussed in Chapter 3. In those cases the sought-after property was *invariance* to as many image plane transformations as possible

while minimizing the loss of global shape information. Another example that has been often mentioned in this thesis has been the case of object-attached features. We have shown that many simplifications occur when an object recognition algorithm relies on object-attached features. A further example is features that are *perceptually significant* [Low85]. The purpose of this section is to examine what characterizes a good feature for the task of *hypothesis generation.* In the context of Cyclops, we will consider what are the best features to use as indices into the database of model instances. Of course, we will not require that such features be object-attached

In Cyclops features are considered to be solely 2-d entities. Features may be *detected* directly from the image. In this case they are called *image features* or *observed features.* Conversely, features may be *predicted* using a model, transformation, and scene information. We call such features *model-generated features* or *predicted features.* Unlike many other 3-d recognition systems, Cyclops does not employ the object-attached feature assumption in any way. This is accomplished by not giving any 3-d interpretation to the features and only using their well-defined 2-d interpretation. This is not to say that the features do not have a 3-d interpretation. Indeed, each feature is the result of imaging some set of surface points in the 3-d scene. However, for most objects and scenes, the particular surface points that are imaged change in a difficult to predict manner as the viewpoint is varied, rendering the 3-d interpretation of the features difficult to use during the hypothesis generation phase of Cyclops.

4.3.1 Properties of Good features for 3-d Object Recognition

Given that we have required that features be 2-d, what properties should they possess to facilitate the hypothesis generation process? There are several:

- spatial locality,
- selectiveness,
- perceptual significance,

- noise resistance,
- insensitivity or invariance with respect to variations in insignificant viewing parameters,
- smoothness with respect to variations in significant viewing parameters,
- ease of segmentation, and
- representational compactness.

We discuss each in turn.

4.3.1.1 Spatial Locality

Spatial locality is an important property as it allows recognition to be performed in domains where noise and encroachment of the background may cause parts of the object to be distorted or to be missing in the image. Spatially local features have a smaller chance of being distorted since a small feature stands a good chance of being on a portion of the object that is undistorted. In addition, recognizing partially obscured objects requires spatially local features for the same reasons.

It is useful to extend the idea of spatial locality to include certain features that may, in fact, be derived from portions of the image that are widely separated. To do this, we first say that a feature is *strictly spatially local* if the pixels in the image that went into computing it were contained in a small region of the image. A feature can then be considered to be spatially local, though not strictly, if it is comprised of features that themselves are strictly spatially local. Features that are spatially local in this extended sense permit recognition of objects that have been distorted by noise, encroachment, or occlusion.

Spatially local features may be referred to as *primitive* or *compound* features. A *primitive* feature is detected from a spatially local portion of the image, while a *compound* feature may, in the general case, be composed of a number of primitive features, or,

recursively, other compound features. For example, an inflection point, i.e., a point of zero curvature on an edge contour, is an example of a primitive feature. A compound feature might be a trio of such points. The attributes of such a compound feature may consist of the 2-d spatial relationships among its component inflection points. Whether it be compound or primitive, any feature that is part of another feature is a *subfeature* of its parent.

Compound features should not be confused with global features as they are considerably different. In using a global feature, we seek to encapsulate the shape of an entire object. In contrast, a compound feature may record the shape of an object where the primitive features that comprise it are located, as well as the spatial relationship between the primitive features. Thus, a typical compound feature is a far less complete description of the image than a global feature. However, for the purposes of hypothesis generation, this level of description is usually sufficient.

4.3.1.2 Selectiveness

Selectiveness is a key property. A feature is *selective* if a set of model instances that contain the feature is small, i.e., the presence of a selective feature largely determines the model identity and viewing parameters.

Selectivity is dependent on the system's vocabulary. For example, a pair of line segments may not be very selective if the vocabulary consists of polyhedral objects. However, a pair of line segments may be very selective if the vocabulary consists of smoothly curving objects and one polyhedral object.

Selectivity is analogous to the *salience* of features for 2-d objects, as defined by Turney [Tur86]. In essence, Turney's salience measure is the inverse of the number of times a segment of a 2-d model's boundary matched other model segments in the system's vocabulary. A segment that was not very salient would have many other matching segments. If detected in the image, such a feature would generate many hypotheses. On

the other hand, a salient, or selective, feature would match few segments in the vocabulary of 2-d models. Therefore, detecting such a feature would lead to the generation of few hypotheses. Turney's thrust was to find features that were maximally salient, and search for such features in the image. If discovered, such features provided powerful evidence of the presence of the object.

While Turney demonstrated that quantitative determination of the selectivity of 2-d objects is possible, although computationally expensive, extending his approach to 3-d is not possible. Nevertheless, it is important to be able to determine the types of features that are likely to be selective. In this regard, Turney's work provides an interesting insight: the most salient, or selective, boundary segments of 2-d models were most often those possessing regions of high curvature. An example of this is illustrated in Fig. 4.10.

While the importance of curvature to perception, and object recognition in particular, is not in doubt, there is controversy concerning whether the points of maximal curvature or points of significant changes in sign of curvature are more important. Attneave [Att54] initiated the idea that curvature maxmima are perceptually important. His evidence for this was twofold. First, he produced a line drawing of a cat where the endpoints of the lines fell on points of high curvature. Attneave's cat is shown in Fig. 4.11. The fact that most people easily recognize the cat was given as evidence that high curvature points are perceptually important. Second, Attneave asked subjects to place ten points on a curve in order to achieve the best approximation to it. Naturally, people placed the dots near points of high curvature, and Attneave concluded again that high curvature points hold a special place in human perception. However, both of Attneave's conclusions are suspect. In response to Attneave's first claim, Lowe produced a redrawing of Attneave's cat, seen in Fig. 4.12, where the high curvature points have been moved to the center of the line segments in the original, as far as one can get from the original locations. Nevertheless, the cat remains as easily recognizable and similar in appearance to the original, although Lowe's drawing is likely a less accurate approximation to the original smooth curves that

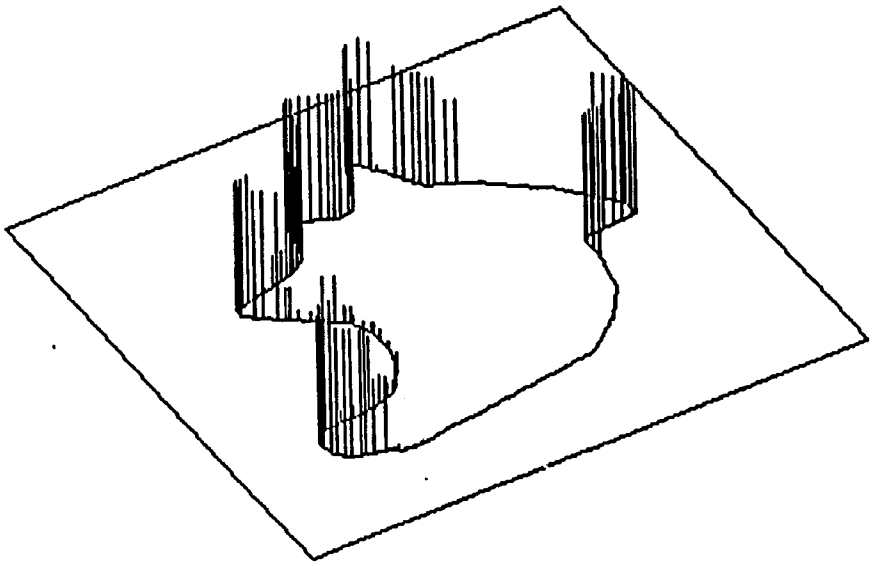


Figure 4.10. Above is a perspective view of the contour of a 2-d door-lock part. Extending from evenly spaced points on the contour are vertical bars whose lengths are proportional to the saliency, or selectiveness, of the contour neighborhoods which have the points as their centers. Salience is the inverse of the number of times the contour segment of interest matched sufficiently well in the object set. Note the large correlation between the amount of high curvature boundary that is contained in the segment and how informative it is. This figure was reproduced from Turney [Tur86].

comprised the cat. In addition to refuting Attneave's arguments, Lowe used his cat to strengthen his case for using line segments in place of high curvature points as features in his own recognition system. Attneave's second experiment, where people were asked to place points on a curve, reflects the fact that, when approximating a curve with line segments, placing their endpoints near to high curvature points will result in a good approximation to the curve.

In the same vein as Lowe, Huttenlocher [Hut88] has argued that *inflection points* and *line segments*, not high curvature points are likely to be the most selective features. He based his conclusions on the fact that inflection points and line segments are "invariant

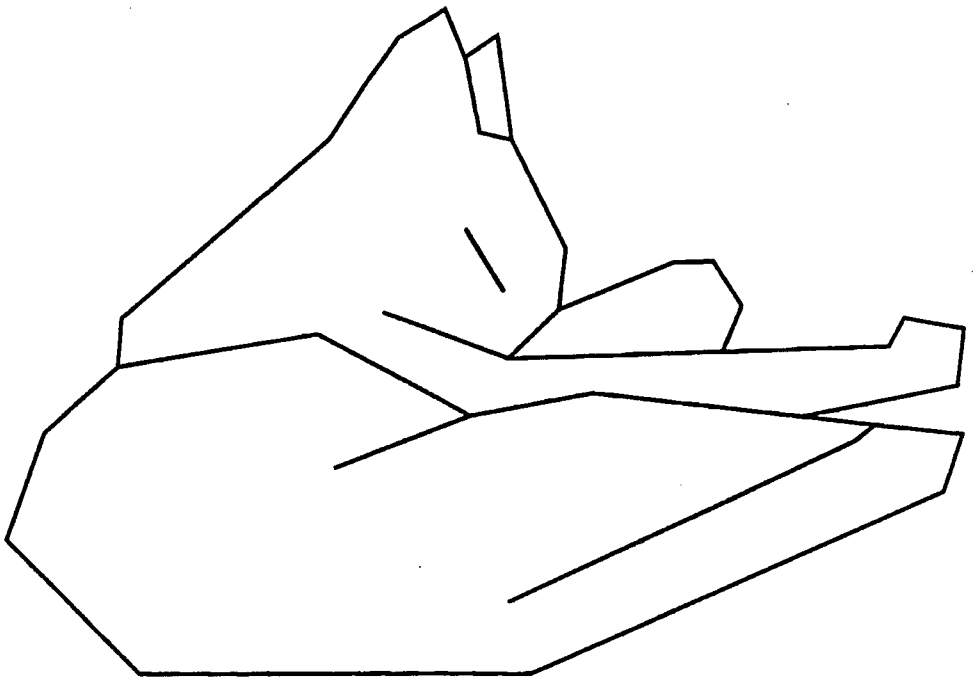


Figure 4.11. Attneave's drawing of a cat. The endpoints of the line segments are located at high curvature points on the cat's boundary. The cat's easily recognized form was supposed to be evidence for the perceptual importance of high curvature points.

to projection" (that is, they are object-attached features) and looked to Lowe's cat for additional support for his argument. This reasoning is flawed, since, on objects not possessing creases, inflections are *not* invariant under projection.

The types of features being most selective for the human vision system remains a topic of active research. However, this question does not bear directly on machine recognition. For all his preoccupation with human vision, Lowe points out that there are many image relationships that the human vision system is poor at detecting, but would be very useful to a machine vision system because they are highly selective or, referring to the next section, perceptually significant. Thus, for machine recognition, perhaps Turney's work provides the most insight. His work indicates that for 2-d objects, high curvature points and inflection points are both very selective features. Of course, Turney's work also indicates that single line segments are very unselective features. As this observation

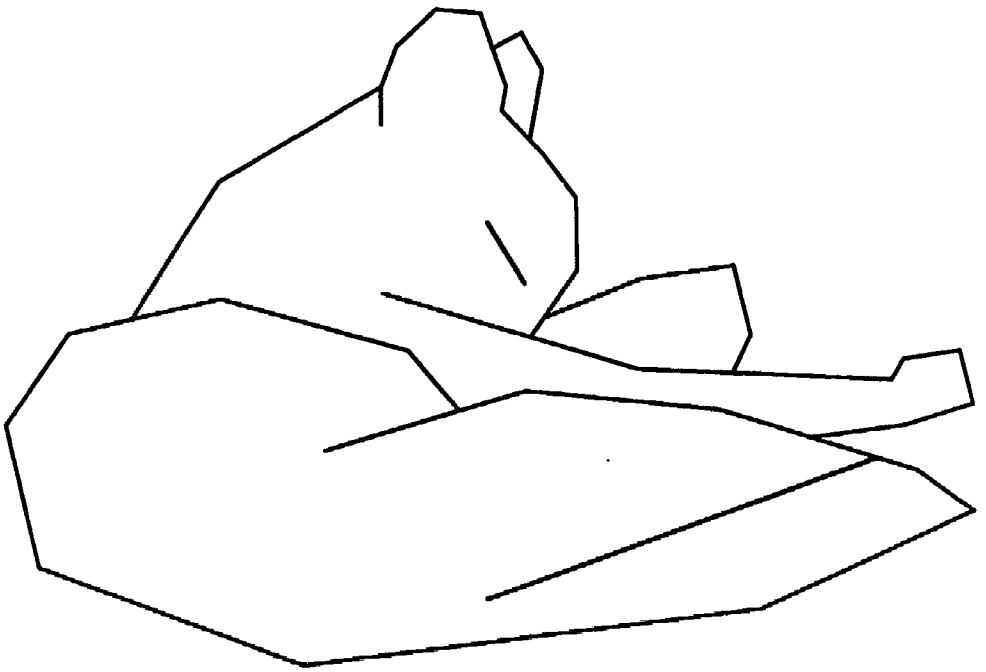


Figure 4.12. Lowe's redrawing of Attneave's cat. The endpoints of the line sements were moved to the center of the segments of the original cat, i.e., as far as possible from the original locations of the high curvature points. Nevertheless, the cat remains easily recognizable. Lowe used this to refute Attneave's claims as to the perceptual importance of high curvature points, as well as to bolster his arguments that inflection points and line segments are as perceptually important as high curvature points.

would lead us to expect, both Lowe and Huttenlocher, and others who employ line segments, produce selective features by grouping line segments into *compound features*. We will return to the fitness of high curvature points, inflection points, and line segments as features as we discuss the remaining properties of a good feature.

We alluded in the previous paragraph that it is possible to build selective compound features from unselective primitive component features such as line segments. This is important, since primitive features such as high curvature points, inflection points, and line segments typically are not selective enough individually to provide a large reduction of the search space. That is, recalling the discussion of backprojection in Chapter 3, the backprojection of such simple features yields large regions in model-instance space.

If the effects of noise are ignored, each independent image measurement encoded in a feature reduces the dimensionality of the backprojected volume in model-instance space by one. For example, a directed inflection point feature possessing the attributes x location, y location, and orientation would reduce the dimensionality of the space to be searched by three. Two such features would allow six dimensions to be determined. As discussed in Chapter 1, model instance space consists of six continuous dimensions and one discrete dimension. Thus, in the absence of noise, a feature built from a pair of directed inflection points would backproject to a discrete and finite number of points in model-instance space. This conclusion is supported by Huttenlocher [Hut88], who used pairs of directed inflection points to calculate the pose of a model given a correspondence between a pair of directed 3-d points and 2-d directed inflection points. In reality, the attributes of the directed inflection points contain noise. In this case, discrete points that originally comprised the backprojection of the feature spread into volumes, reducing the selectiveness of the compound feature. In this case, adding primitive features to the compound feature may be necessary to make the feature selective enough to be useful.

Increasing the selectivity of features by making them more complex is not without cost. Given a set of primitive features as possible components of a compound feature, the number of compound features that could be made from this set grows combinatorically. It is possible that many, if not most, of the possible compound features may be the result of primitive features detected on distinct objects or generated by noise. Unless it is possible to select groupings of primitive features that are likely to have been generated by the same object in the system's vocabulary, making features more selective by increasing their complexity will ultimately backfire since the system will lose more than it gains due to the necessity of processing a combinatoric explosion of many features, even though the resulting features may be more selective.

4.3.1.3 Perceptual Significance

A perceptually significant feature is unlikely to have arisen at random in the image, either by noise or by an accident of viewing. Equivalently, such features are *significant* to the recognition process because they are likely to indicate the presence of an object. The perceptual significance of a feature can be defined as the probability that an object in the vocabulary appears in the scene given the appearance of a feature. This can be empirically determined from a set of representative images. Calculating it from first principles is generally not feasible as it depends on the both the type of backgrounds that will appear in the images as well as the nature of the objects in the vocabulary.

Lowe [Low85] has placed a great deal of emphasis on perceptual significance in the recognition process. His definition of significance is somewhat more vague than ours: "...the probability that they [the features] are non-accidental in origin", i.e., the probability that an instance of a feature "arose for a causal reason". In order to construct perceptually significant features, that are also sufficiently selective, Lowe describes a method for forming perceptually significant groupings of linear segments. These groupings are based on image relations such as proximity, colinearity, and parallelism, although the last two are of limited use when non-polyhedral objects may appear.

Complex scenes may contain many compound features. Many of these features may be meaningless, and absorb valuable computation to determine their true significance. If compound features can be ranked by perceptual significance, much computation can be saved by processing the most perceptually significant features first. This is Lowe's primary thrust: since the groupings of line segments in his system, SCERPO, could consist of several line segments, it would have been impossible to examine all possible combinations. However, with heuristics for perceptual significance based on proximity, parallelism, and colinearity, SCERPO was able to greatly reduce the number of features that the system needed to examine, making it feasible to use these large groupings of line segments. Since the groupings were so large, the features tended to be selective as well.

Lowe relied on the observation that in most 3-d scenes containing polyhedral objects, line segments that have endpoints in proximity, nearly parallel, or nearly collinear are unlikely to have been generated by an accident of viewpoint. Rather, such relationships among the image features are likely to reflect particular 3-d structures. For example, if two line segments nearly coterminate, then it is likely that the line segments are the result of imaging two linear creases that intersect at a 3-d vertex. However, the relationships of parallelism and colinearity become much less useful when objects are smoothly curving and may possess few, or any, linear creases. Fortunately, proximity remains a valid relation for determining perceptual significance for smoothly curving objects. However, by itself, proximity is not very useful for grouping primitive features into perceptually significant compound features. There is another, very general, property that can be used to assess perceptual significance: *continuity*. Continuity in any image property may be used to determine perceptual significance. For example, two primitive features that are detected on a portion of edge contour that is *continuous*, i.e., possessing no breaks, grouping such primitive features into a compound feature results in a far more perceptually significant feature than two primitive features that are derived from two unconnected edge contours. The continuity in properties of regions between two contours can also be used to determine where features derived from the two contours would be perceptually significant [Hut88].

4.3.1.4 Noise Resistance

Noise resistance is a critical property since it impacts both the selectiveness and the perceptual significance of a feature. For example, if it were possible to compute a noiseless feature, then the feature could be perfectly selective, assuming that it is complex enough. The reason for this is that the backprojection of such a feature into model-instance space would be either an impulse of probability, allowing the model identity and viewing parameters to be determined with complete certainty, or identically

zero everywhere, indicating a feature that cannot be generated with the system's current vocabulary. Conversely, if the feature is very noisy, then many model instances will be significantly probable when the feature is backprojected. By definition, such features are not selective.

Perceptual significance is similarly affected by noise. If a feature is not noisy, then the continuity of image properties can be ascertained accurately, allowing the perceptual significance to be computed. On the other hand, if the feature is noisy, then the system may not be able to determine continuity and proximity reliably, making estimates of perceptual significance unreliable.

4.3.1.5 Insensitivity or Invariance with Respect to Variations in Insignificant Viewing Parameters

It is often possible to compute the attributes of features so that variations in certain viewing parameters are not reflected by variations in the attributes of the feature. Such features are *invariant* or *insensitive* to these viewing parameters.

Invariance and insensitivity to viewing parameters, such as scene lighting, that have no impact on the goal of the recognition system to identify and locate objects in an image, is clearly advantageous. Even viewing parameters that must eventually be determined may be unimportant during the hypothesis generation process. Invariance or insensitivity to such parameters reduces the dimensionality of the search space. In addition, if certain additional, non-invariant attributes are retained, the values of such parameters can usually be recovered. As an example, the best possible situation would be to compute a feature that is invariant to variations in all of the parameters of the viewing transform. Although such features do not exist for intensity images, they may be computed from range images, using, for example, 3-d moment invariants. Such range features would allow models to be identified by matching the invariant feature attributes. The 3-d viewing transform

could then be computed using knowledge of the model and the range data that went into the computation of the 3-d moments.

As explained previously, it is generally not possible to form features from intensity data that are invariant to all six parameters of the viewing transform unless object attached features are assumed [Wei88]. Rather, under weak perspective, invariance to image plane rotation, translation, and scaling is all that can be achieved. Invariance to out-of-plane rotations is not possible. Under full perspective, invariance to image plane rotation and translation is possible. Features can be formed that are insensitive, but not invariant, to translation along the line of sight. As in the case with weak perspective, invariance to out-of-plane rotations is not possible.

Clearly, perfect invariance is never achieved in the presence of noise. In many cases, such as high order moment invariants, noise overwhelms the differences between the values of the feature, rendering it completely unselective, and, therefore, useless for recognition.

4.3.1.6 Smoothness with Respect to Variations in Continuous Viewing Parameters

If a feature's attributes vary rapidly as the real-valued viewing parameters are varied, this indicates that the feature is ill conditioned in the sense that a small change in the value of a features attribute may result in drastic changes in the backprojection of the viewing parameters. Such features are very susceptible to noise. Most "raw", i.e., directly measured, image relationships possess this smoothness property naturally. However, when such relationships are processed, for example, to form an invariant feature, artifacts may be introduced. Care must be taken to avoid this.

4.3.1.7 Ease of Segmentation

This is a property that is often overlooked by researchers in object recognition. A feature that appears to be excellent may be beyond the capability of low level vision modules to reliably detect. A classic example is the features of blocksworld domains which, even today, remain largely beyond the capability of low level vision modules to reliably detect.

4.3.1.8 Representational Compactness

Representational compactness is desirable simply because it reduces the amount of data the system must process.

4.3.2 Features for Hypothesis Generation in Cyclops

The above discussion has clarified the requirements for a good feature. Since Cyclops is an edge based approach, the features we use are computed from edge contours. For primitive features, we have chosen to use the shape of edge contours in the neighborhood of high curvature points and inflection points. With this choice, we sidestep the controversy over the relative merits of high curvature points and inflection points. We believe that both high curvature points *and* inflection points are selective and perceptually significant primitives. We believe that line segments would be a worthwhile addition to the list of primitive features, simplifying the recognition of object that possess many linear features. However, we have not investigated this.

The question now arises: are these primitive features sufficiently selective and perceptually significant to use them individually rather than forming compound features? For example, if the size of the neighborhood about the high-curvature point and the inflection point is made larger, the feature should become more selective. In addition,

since the feature is derived from a single portion of an edge contour, it is naturally perceptually significant. Further, avoiding the necessity of forming compound features, many of which may be meaningless, yet absorb valuable computational resources, is attractive. Unfortunately, our experience leads us to conclude that such features suffer from the problem that, if they are made large enough to be sufficiently selective and they are made invariant to image plane rotation, translation, and scaling, then they become extremely sensitive to out-of-plane rotations. We conclude that it is generally necessary to form compound features from the primitives, and, if necessary, rely on heuristics for estimating perceptual significance to reduce the computation wasted on generating and processing meaningless features.

Using a pair of primitive features has proven to yield very selective features. Fig. 4.13 shows a schematic of the features used by Cyclops' for hypothesis generator. The curves E_a and E_b are edge contours. Points a_0 and b_0 may be high curvature points, inflection points, or a mixture of both. Point c is the center of line a_0b_0. If l is the length of a_0c, then points a_1 and a_2 are located a distance αl on either side of a_0 along the curve E_a, where α controls the degree of spatial locality of the primitive features. The invariant attributes of the feature are:

- the normalized distances $d_n^a \equiv d(c, a_n)/l$, $n = \pm 1$, and where $d(c, a_n)$ is the distance between points c and a_n;
- the normalized distances $d_n^b \equiv d(c, b_n)/l$, $n = \pm 1$, and where $d(c, b_n)$ is the distance between points c and b_n;
- the orientations o_n^a of the segments ca_n with respect to segment ca_0, $n = \pm 1$;
- the orientations o_n^b of the segments cb_n with respect to segment cb_0, $n = \pm 1$;
- the angles t_n^a between line ca_0 and the tangent to E_a at a_n, $n = 0, \pm 1$;
- the angles t_n^b between line cb_0 and the tangent to E_b at b_n, $n = 0, \pm 1$.

The orientations range over $[0, 2\pi]$. The angles are confined to $[0, \pi]$ since the tangent lines are undirected. The reason for this is to account for possible contrast reversals along the contours E_a and E_b. The orientations, tangent angles, and normalized distances all

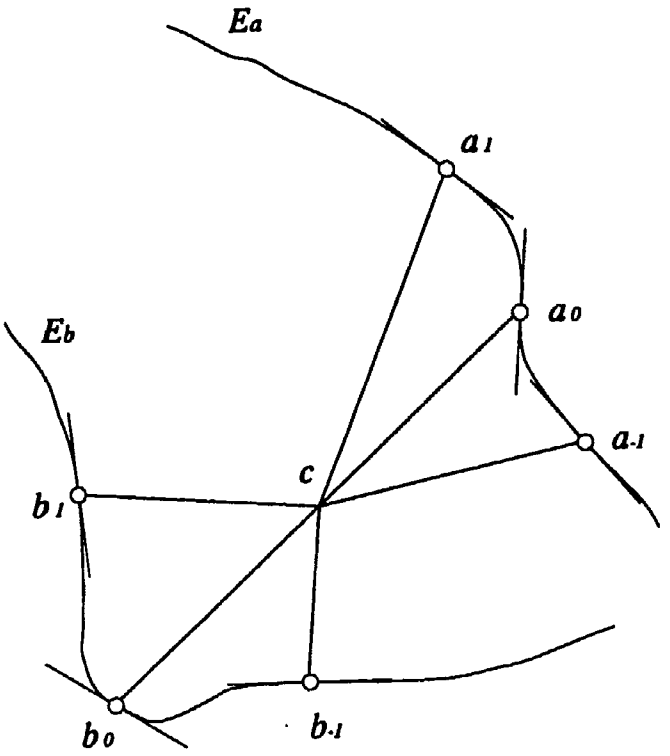


Figure 4.13. A point-pair feature. The primary points of the feature, a_0 and b_0, may be any combination of inflection points or high curvature points. The auxiliary points, $a_{\pm 1}$ and $b_{\pm 1}$, are derived by traversing a length of edge contour, αl, starting from a_0 or b_0, where l is the length of the line segment a_0b_0, and α is a proportionality constant. At each point, the tangents to the curve are calculated as well as the lengths and relative orientations of the segments from the center of a_0b_0, c, to the points. Calculating the invariant attributes of this feature is described in the text.

encode information about the local shape of the curves E_a and E_b in the neighborhood of points a_0 and b_0.

That these attributes are invariant to image plane transformations is not difficult to demonstrate. Assume that E_a and E_b are part of the same object. Then, if the object is translated, none of the attributes change since all measurements are taken relative to the center point c. Similarly, if the object is rotated about c, none of the feature's attributes change. The distances do not change since it is a rigid rotation, and the angles do not change since all measurements are relative to the centerline a_0b_0. If the object is

scaled by a factor, say *beta*, then all of the distance measures are scaled by β as well. The attributes are formed by ratios of distances, and the common scale factor cancels. Thus the attributes are invariant to scale changes as well. Of course, all of the angular measurements are invariant automatically.

Other, non-invariant attributes of the feature include:

- the coordinates of c;
- the orientation of a_0b_0;
- the length l of a_0b_0.

These attributes allow the image plane transformation parameters to be computed given a matching model feature.

4.3.2.1 Detection of Point-Pair Features from Images

The steps involved in detecting point-pair features from an image are summarized below:

1. Detect edges.
2. Link edges into edge contours, filtering them by length.
3. Compute *x-y-slope* versus arclength representation of edge contours.
4. Detect significant high curvature points (hereafter *critical points*) and inflection points.
5. Estimate the direction of the tangent line at the critical/inflection points and the auxiliary points.
6. Using the position and estimated slope of the edge contour at the critical/inflection points and at the auxiliary points, compute the attributes of the point-pair features.
7. Filter the point-pair features based on perceptual significance.

We now describe each step in greater detail.

4.3.2.2 Edge Detection and Linking

The feature detection algorithm starts with an image, such as Fig. 4.14. Canny's edge detector [Can83] is then applied to the image to obtain a list of the locations and directions of edge pixels. The result of the edge detection is shown in Fig. 4.15. The pixels are then linked into raw edge contours by a simple linker. The linker searches the pixels surrounding the current pixel for another edgel. It does this by starting at the pixel that is most nearly in the direction of the edge indicated by the edge detector, and proceeding progressively to the pixels that are further from this direction until all neighboring pixels have been examined. If an edgel is found, it is added to the ordered list of edgels and edgel directions that comprise the raw edge contour. Then the new edgel becomes the current edgel, and the process is repeated. The algorithm enforces two-connectedness of the contours. Following this step, very short contours that are unlikely to be of any use to the recognition algorithm are removed by passing them though a length threshold. The result of running the linker is shown in Fig. 4.16.

4.3.2.3 A Uniform Arclength Parameterization

We now have a set of raw edge contours that are parameterized by the number of edgels along the contour. This parameterization is not very useful since it is non-uniform, i.e., the actual distance traversed by the contour as a function of the edgel number parameter depends on the direction of the contour. For example, if the contour is diagonally oriented, it will cover 41% more distance per unit of the parameterization than if it is horizontally or vertically oriented. Since it is difficult to accurately determine curvature in the presence of such angular anisotropy in the parameterization, we must calculate a uniform arclength parameterization of the curve. The representation is a dual representation consisting of $\mathbf{r}(s)$, the Cartesian coordinates of the pixel at arclength s, and $\theta(s)$, the angle of the tangent line to the contour relative to horizontal at s. This dual

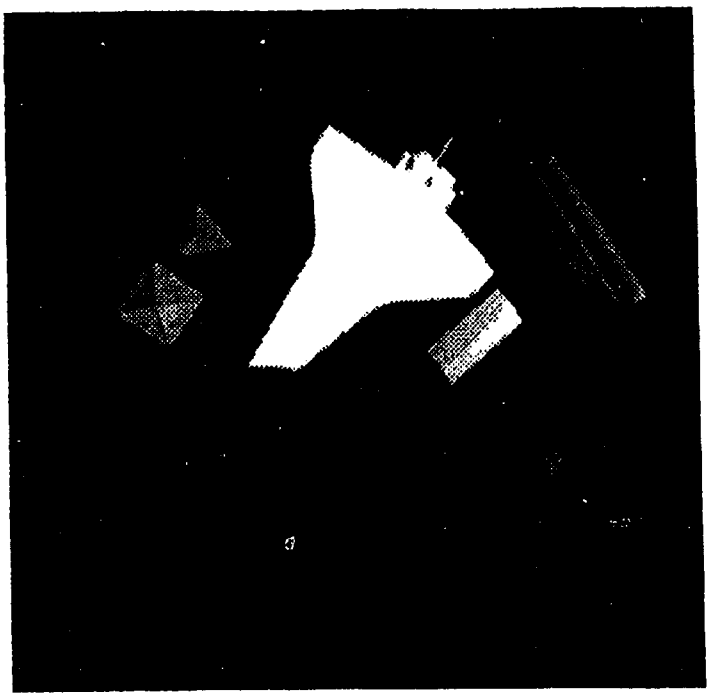


Figure 4.14. An example image used to illustrate computation of features in the text.

representation is useful since it facilitates the detection of the critical points and inflection points. We will refer to this dual representation as the (x, y, θ)-s representation. Other workers who have used the $\theta(s)$ representation of edges include Perkins [Per77], Yam *et al.* [YMA80], Mckee *et al.*, [MA77], Turney [Tur86] and Tsui *et al.* [CT89].

The (x, y, θ)-s representation described in the previous paragraph is dual in the sense that, under ideal circumstances, either one can be computed from the other given some initial conditions. In practice, however, computing $\theta(s)$ from $\mathbf{r}(s)$ is difficult to do accurately as well as being computationally expensive [Sha85, SA86]. Going from $\theta(s)$ to $\mathbf{r}(s)$ is simple, but computationally expensive. However, it is a simple matter to compute both $\mathbf{r}(s)$ and $\theta(s)$ using the information stored in a raw edge contour. Each edgel of the raw edge contour contains the edgel's Cartesian coordinate as well as the direction of the edge contour at the edgel. This information is a byproduct of edge detection. The algorithm simply computes the points of the uniform arclength parameterization (most

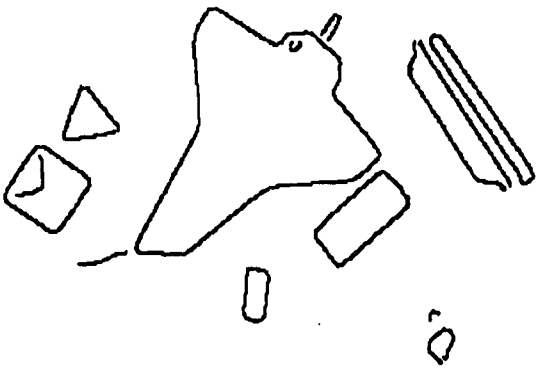


Figure 4.15. Result of applying Canny's edge detector to image of Fig. 4.14.

of which fall between sample of the raw edge contour representation) and does a linear interpolation of $\mathbf{r}$ and θ using the nearest samples in the raw edge contour representation. Referring to Fig. 4.17, the uniform arclength sample points are found, conceptually, by treating each sample in the x-y plane as a "pin" and, starting with the end of the raw edge contour, putting a taut "string" in contact with the pins. The length of the string is the distance between sample points of the (x, y, θ)-s representation. The end of the string is the next sample point, and the values of $\mathbf{r}$ and θ are computed here. This point is also used as the beginning of the string to find the next sample. This process is continued until the raw contour ends. Thus, the distance between samples is constant in a piecewise linear manner. Fig. 4.18 plots separately the x-y and the θ-s parts of the dual (x, y, θ)-s representation for a puzzle piece which has been processed as described above.

Figure 4.16. The result of linking edges and filtering the resulting edge contours by length.

4.3.2.4 Detection of Critical Points and Inflection Points

Using the (x, y, θ)-s representation, it is possible to detect critical points and inflection points. Unfortunately, due to noise and quantization effects, the location of critical points and inflection points depend on *scale*. Scale refers to the size of the operators that are used to detect the critical points and inflection points. Of course, the edge contours themselves are functions of the scale of the edge operators. A treatment of the behavior of edge contours in scale space, i.e., as a function of scale, can be found in [Lu88]. In our case, we use the smallest edge operators possible in order to avoid distorting the true shape of the edge contours. Typically, $\sigma = 1.5$ pixels in the edge detector. This results in many spurious contours, as well as increasing the likelihood that a long contour may be broken. However, these problems are dealt with by the contour grouping module described in the following section. The problem of the scale of the operators to use for

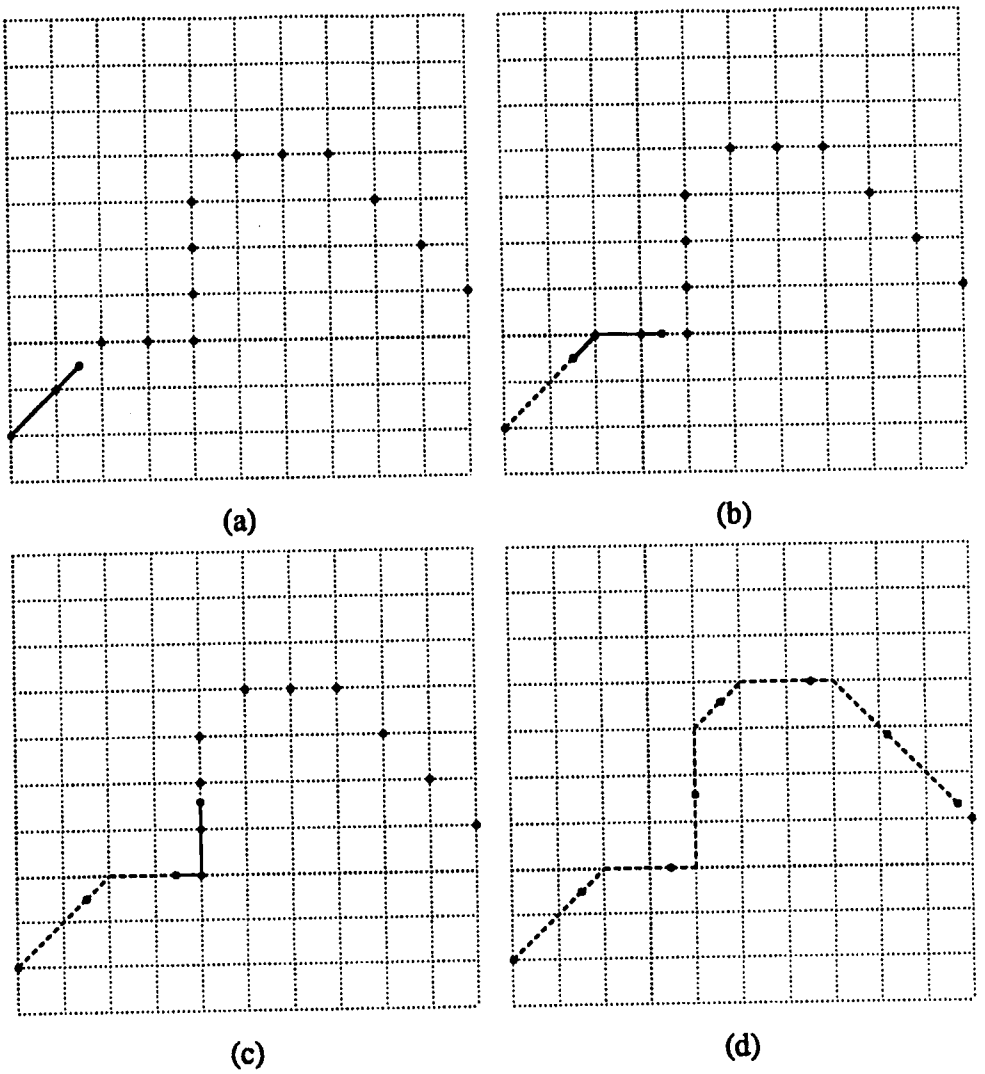


Figure 4.17. Illustrations (a)-(d) show the process used to calculate uniform arclength representations of the edge contours. Black dots denote the original, non-uniform, samples. The uniform resampling of the edge contour is accomplished conceptually by treating the length of arc, ds, desired between each of the sample points in the uniform resampling of the curve a string of length ds. Starting with the initial point, shown as a gray centered dot, the "string" is tautly stretched around the black dots, which can be thought of as "pins". The $\times$ centered dot at the other end of the string is the new sample point. The x, y, and θ values are obtained at the intermediate point by linear interpolation. This process is continued, as shown in (b) and (c) until the entire contour is resampled, as shown in (d).

(a) (b)

Figure 4.18. In (a) is shown the Cartesian part of the (x, y, θ)-s representation of an object, and (b) shows the θ-s part of the (x, y, θ)-s representation of the same object.

detecting the critical points and inflection points remains.

Two different approaches were used to handle the problem of scale. For critical points, only those critical points that are significant over a *range* of scales are kept and used. In order to detect critical points that appear over a wide range of scale, we have developed a multiscale critical point detector which is described in the following paragraphs. For inflection points, the significant inflection points at each scale were kept and used. The detection of inflection points at each scale follows the method of Huttenlocher [Hut88]. For completeness, this method will be summarized in the following paragraphs as well.

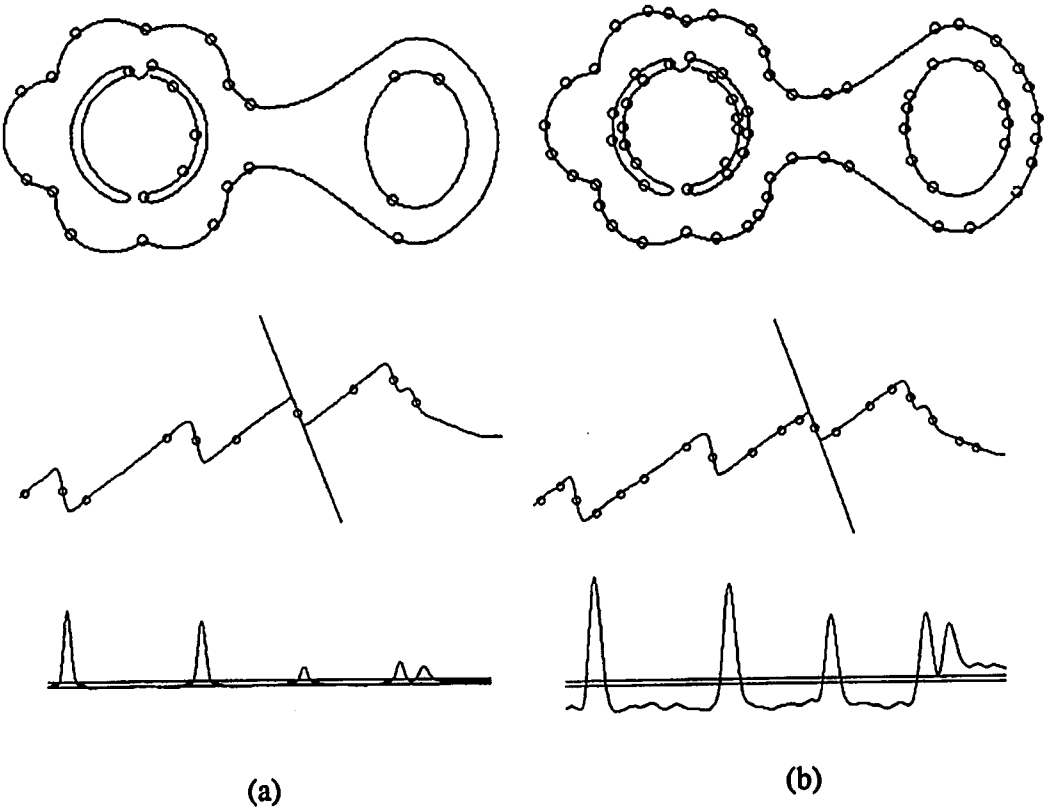
Multiscale Critical Point Detection

We define critical points to be points of locally maximum absolute curvature on the edge contours of objects. From calculus, the definition of curvature is simply the

derivative of the $\theta(s)$ curve with respect to arclength. In practice, differentiating the $\theta(s)$ curve and searching for locally maximal absolute values is not robust since the derivative operator amplifies high frequency noise. Instead, we use a multiscale version of a 1-d derivative of a Gaussian edge detector to detect the critical points that are significant over a wide range of scales. This operator suppresses high frequency noise. The approach is related to the multiscale, 2-d edge detector described in [Sch86]. The use of multiple scales has two major advantages:

- The critical points are stable over a wider range of scale than with a single scale detector.
- With a single scale detector, the localization of the critical point degrades as the width of the Gaussian increases, while the signal to noise ratio degrades with decreasing width. With our multiscale method, the localization of the edge detector is nearly as good as the smallest scale, but has the noise immunity of the largest.

The multiscale critical point detector operates in the following manner. First, the original $\theta(s)$ curve is convolved with a progression of l Gaussian derivatives, each having a σ_i such that $\sigma_i = \rho\sigma_{i-1}$, where ρ is the ratio between two scales. The resulting set of curves, $\alpha_i(s), i = 0...l$, have maxima where the rate of change of the slope angle is large at the scale of the particular Gaussian derivative that was used to obtain it. The l curves are then multiplied point by point to yield a curve $\beta(s) = \prod_{i=1}^{l} \alpha_i(s)$ whose local maxima we will define as critical points. Peaks that appear in a few of the $\alpha_i(s)$ will be amplified in $\beta(s)$ with respect to noise, since it is unlikely that noise peaks will appear in more than one of the $\alpha_i(s)$. In addition, the smaller scale curves will define the width of the peaks in the $\beta(s)$ curve, providing the best localization of the critical points. For more details on the advantages of this method, see [Sch86]. As an illustration of the above method, Fig. 4.19 shows the result of applying the multiscale critical point detector to an actual object, a baby's rattle. The results of using three scales having ratios



(a) (b)

Figure 4.19. Shown above is the result of applying the multiscale critical point detector to an image of a baby's rattle. On the left is the sequence of curves that results from using three scales, with $\sigma = 4$, 6 and 10 samples respectively. On the right is the result of applying a single scale detector to the same edge contour. The sequence of curves, from top to bottom, are: the x-y representation of the contour with detected critical points shown; $\theta(s)$ with detected critical points shown; $\beta(s)$. Note the considerably larger amount of noise present in the single-scale $\beta(s)$ curve which results in many spurious critical points being detected.

of 1.5 and a single scale are shown. The advantages are most apparent in the nature of the $\beta(s)$ curve. There is much more noise in the single scale detector.

Detecting Inflection Points

An inflection point is defined as a zero of curvature. For the same reasons described above, a derivative of a Gaussian operator was convolved with the $\theta(s)$ portion of the (x, y, θ)-s representation to obtain an estimate of the curvature of the edge contour. Following Huttenlocher [Hut88], the *significant* zero crossings of the curvature function are found by examining pairwise all adjacent zero crossings of the curvature function. If the area under the curvature function is greater than a threshold, then the pair of inflection points is deemed significant. This is shown in Fig. 4.20. The area can be can be easily calculated since it is the integral of the curvature function from one zero crossing to the next. Serendipitously, since $\theta(s)$ is the antiderivative of the curvature, the area is simply the magnitude of the difference of the $\theta(s)$ at each of the two zero crossings.

The area method seems to work better than other methods of detecting significant zero crossings, such as requiring the slope of curvature function at the zero to be large enough, or requiring that the peak of the curvature between the two zeroes be sufficiently large. Watt and Morgan [WM85] evaluate several methods for thresholding zero crossings. They found that the moment-based methods, such as the area method described above, to be superior to the other methods.

The scale of the inflection points is determined by applying the derivative of a Gaussian operator to the (x, y, θ)-s contour, thus obtaining the curvature function. Unlike the case for critical points, no effort was made to find the inflection points that are significant over a wide range of scale. Rather, all of the significant inflection points detected at each scale are kept for use by the recognition processes.

4.3.2.5 Estimating the Tangent Direction at Points Along the Edge Contour

Computing the attributes of point-pair features requires both the position and slope of the edge contours at the critical/inflection points of the feature as well as at the auxiliary

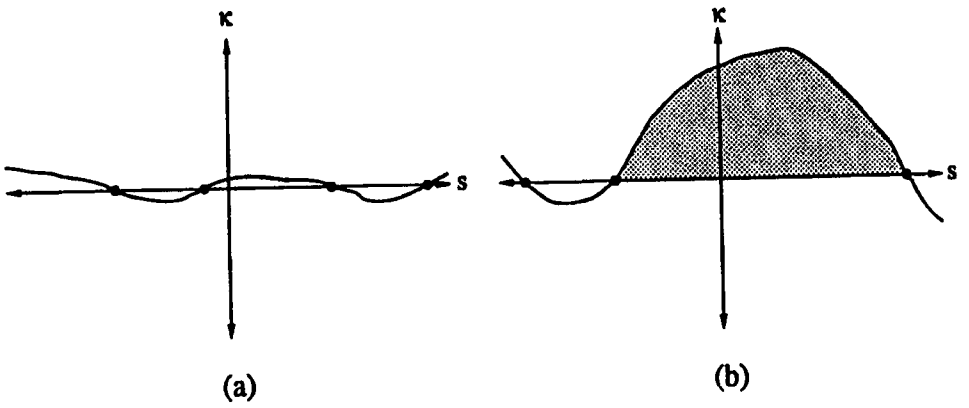


Figure 4.20. The most effective methods of determining which zero crossings of the curvature functions are significant are based on moments of the curvature function [WM85]. We have followed [Hut88] in using the zeroth moment, or area, of the curvature function. A pair of adjacent zero crossings of the curvature function are deemed *significant* if the area under the curvature function exceeds a threshold. Thus, the zero crossings in (a) are significant while those in (b) are likely to be rejected as insignificant. As described in the text, the area under the curvature function is simply the turning angle of the curve, and can be easily obtained from the $\theta(s)$ curve as $|\theta(s_l) - \theta(s_h)|$.

points located equal intervals of arclength on either side of these primary points (recall Fig. 4.13). The position of these points may be simply obtained from the $(x(s), y(s))$ part of the (x, y, θ)-s representation. In principle, the direction of the tangent could be obtained in the same manner from the $\theta(s)$ part of the (x, y, θ)-s representation. In practice, this simple approach is not good as the $\theta(s)$ curve contains noise. Noise in the estimates of the slope of the edge contours will result in some of the attributes of the point-pair feature being noisy, and therefore less selective and perceptually significant. Better results are obtained if several neighboring samples are used to estimate the slope. We have used a Gaussian averaging window whose width is proportional to the distance between the primitive points comprising the point-pair feature. The reason for scaling the size of the window is to prevent the estimate from being dependent on the scale of the

feature. Since the direction of the tangent lines are unsigned, the slope angle is reduced to the range $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$.

4.3.2.6 Computing the Attributes of Point-Pair Features

Computing the attributes of the compound features point-pair features from primitive critical points and inflection points has two areas of difficulty. Both stem indirectly from our need to be able to compare the attributes of point pair features *directly*, attribute by attribute. Our need to be able to compare the features directly by attribute stems from our fundamental assumption that features will be nearby in feature-attribute space if and only if they possess similar structures. Thus, it is important to insure that this property holds.

The first way in which the above property may be violated relates to the ranges of the angular attributes, which, in general may differ by multiples of 2π. This problem is easily dealt with by insuring that all angular attributes are shifted into the same range of possible values. In the case of the orientation attributes o_n^a and o_n^b, the allowed range is $[-\pi, \pi]$. In the case of the tangent angles t_n^a and t_n^b, since the tangent lines are undirected, the allowed range is $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$.

The second possible violation of the above property relates to problems encountered due to the symmetry existing among the attributes of the feature. These symmetries may result in two features possessing totally different attributes even though they describe exactly the same structure. Both of these problems lead to discrepancies in the attributes of features that we desire to possess identical attributes. Since the features are used as indices into the database of model instances, discrepancies in the attributes will lead to failure to generate good hypotheses, thus degrading the performance of Cyclops.

To clarify the nature of this problem, Fig. 4.21 shows the four possible assignments of the points a_{-1}, a_1, b_{-1}, and b_1, assuming that the feature consists of distinguishable primitives. If the feature consists of two identical primitives, i.e., both are critical points

or both are inflection points, then four additional assignments exist corresponding to the possibility that the primitives are reversed. Thus due to the symmetry of the attributes, and depending on the details of the configuration of edge contours in the image, four or eight possible features may result. Thus, without taking further measures, if such features were used to index the database, there would alway be at least a 75% chance that the feature would be missed since the wrong attributes would be indexed. This is unacceptable.

A simple solution to the problem of attribute symmetry would be to include in the database one feature for each possible assignment of primitives. However, this would multiply the number of features by a factor of four to eight. In addition, four to eight times as many hypotheses would be generated, most of which would be invalid. Another solution would be to query the database for all the possible features. This has the same problem as the first possible solution in that roughly four to eight times as many hypotheses will be generated. A preferable approach, avoiding all of these drawbacks, is to put all the features into a canonical representation that would allow the feature attributes to be directly compared. To accomplish this, we first order the points a_{-1} and a_1 so that the point with the smallest normalized distance attribute is assigned as a_{-1} and the larger is assigned to a_1. The same procedure is carried out with b_{-1} and b_1. Next, if the primitives are of the same type, the degree of asymmetry between the normalized distances for each primitive is examined. The degree of asymmetry for primitive a is computed as $|a_{-1} - a_1|$, and similarly for primitive b. If the primitive with the largest asymmetry is not primitive a, the primitives are reassigned so that primitive a is the most asymmetric. If the primitives are not of identical types, then the critical point is assigned as primitive a and the inflection point is assigned as primitive b.

Were it not for noise, the above procedure would always result in canonical features where identical structures would always have the same attributes. If the values of the normalized distances are such that they are near a decision boundary as to how to assign

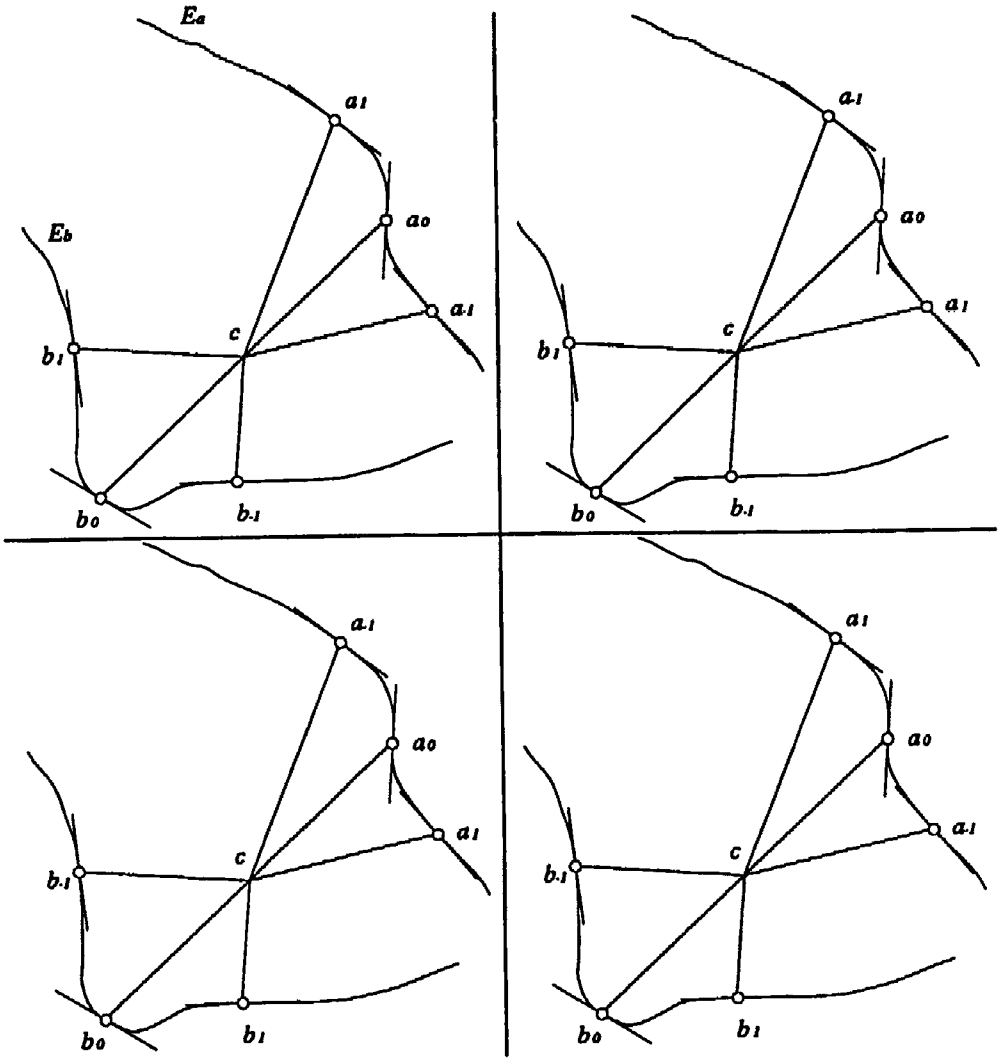


Figure 4.21. Assuming that the primary points a_0 and b_0 can be distinguished, i.e., one of them is a critical point while the other is an inflection point, the figure shows the four possible assignments of the auxiliary points a_{-1}, a_1, b_{-1}, and b_1. Which assignment is actually realized depends on the direction of the parameterization of the edge contours. Each of the possible assignments of the auxiliary points leads to features that are generally distant from each other in feature-attribute space, but which possess visually identical stuctures. We wish to avoid this situation as it defeats the principle of neighborhood searching.

the attributes, then a small amount of noise could result in drastically different attributes. To prevent this, Cyclops uses empirically determined estimates of the amount of noise in

the normalize distance attributes to determine when a feature has an ambiguous representation. If it does, the feature is marked as ambiguous. If an ambiguous feature is to be used to index a model instance, then all symmetric features indicated by the ambiguity are added as well. However, since this situation occurs so rarely, it insignificantly affects the size of the database and the average number of hypotheses generated.

4.3.2.7 Determining the Perceptual Significance of Point-Pair Features

When point-pair features are computed from primitive inflection points and critical points, the point-pair features whose primitive features were detected on portions of edge contour that are derived from different objects in the image are meaningless. Similarly, features that possess primitives that were detected on edge contours generated by noise, unknown object boundaries, or background are also meaningless. Since such features may be the majority in a complex scene, a great deal of computation can be saved if the meaningful features, i.e., the features whose parent edge contours are derived from the same object, could be determined. This information could be used to avoid computing the attributes point-pair features that are likely to be meaningless, as well as preventing false hypotheses to be generated and tested. Hence the algorithm stands to gain considerable efficiency if most of the meaningless features can be eliminated.

The key problem is determining which contours are likely to have arisen from the same object. We have taken a rule-based approach to this problem, since most of our knowledge about the likelihood of contours being from the same object is in the form of heuristics. The system employs proximity cues, as does Lowe's perceptual grouping approach [Low87a], as well as continuity cues. All of the continuity cues in Cyclops are based on contour continuity. In contrast, Huttenlocher's approach [Hut88] employs two kinds of region continuity to group contours.

The contour grouping module takes a list of raw edge contours as input, and the output is a new list of contours, and connection matrix, M. The list of contours is likely to be

larger than the input list because the module often splits contours when there is evidence that two portions of a single contour may not be derived from the same object. Element $M_{ij} \in [0,1]$ of the connection matrix represents an estimate of the likelihood that the i^{th} contour is connected to the j^{th} contour. In the context of this module, two contours being connected is the same as their being derived from the same object. $M_{ij} = 0$ indicates no evidence for connection, while $M_{ij} = 1$ indicates contours i and j are connected with certainty.

The first step in the contour grouping algorithm is to break contours in places where there is the possibility that the simple linking algorithm (described in Section 4.3.2.2) made a mistake by linking contours that are not part of the same object. Such breaks are made where the contours are in close proximity to one another. Breaks could also be made at points of high curvature, although this was not implemented. Then, groups of contour endpoints that are in proximity are clustered into *hot spots*. Each hot spot indicates where the relationship of the participating contours must be determined.

The contour grouping module attempts to classify the relationships between contours into three categories.

1. Continuation: The contours are curvilinear extensions of each other.
2. Join: The contours coterminate, as, for example, at the vertex of a cube.
3. No-connection: The contours bear no meaningful relationship to each other.

These relationships are assumed to be exclusive, and are shown in Fig. 4.22. Joins and continuations are considered as evidence that the contours belong to the same object. Contours not having endpoints that are members of the same hot spot not considered by the algorithm at this time. However, such contours may be connected indirectly, as in Fig. 4.23.

Contours sharing a hot spot may bear continuation relation or join relations to each other in addition to the default no-connection relation. Each possible relation is given a fuzzy certainty value in $[0,1]$. Since the relations are exclusive, they are constrained

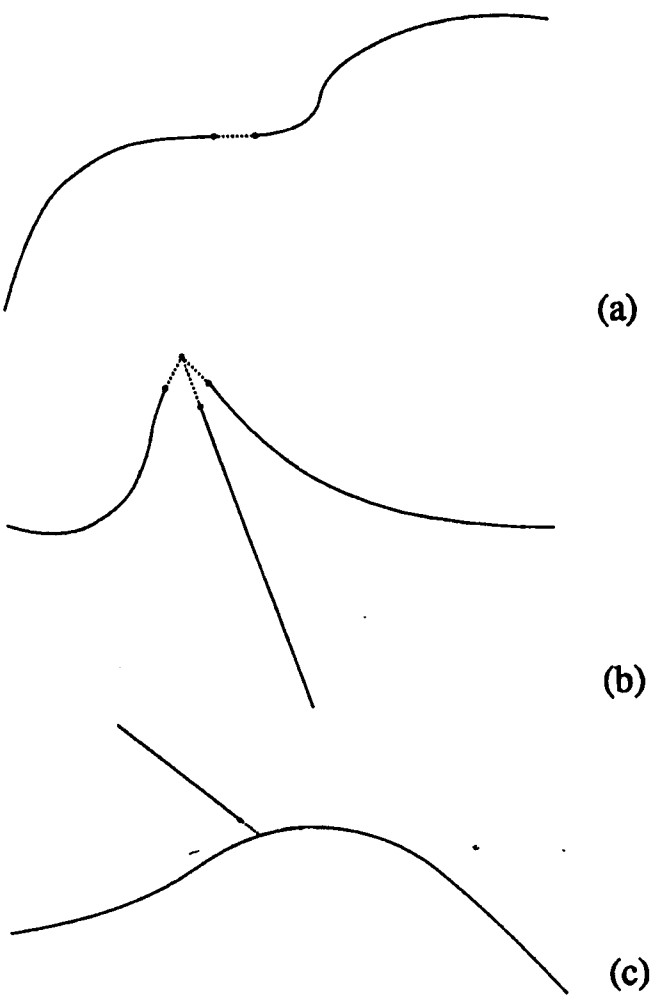


Figure 4.22. (a) shows a Continuation relation. A continuation implies that the participating contours are curvalinear continuations of eachother. (b) Shows a join relation. A join results when the participating contours coterminate. Finally, (c) shows a no-connection relation. A no-connection means that there is no local evidence for either e continuation or join relation.

to sum to unity. This affords them intuitive interpretation as probabilities, although they are not probabilities. The initial values of the relation certainties are set to $\frac{1}{3}$ for no-connection, $\frac{1}{3}$ for join and $\frac{1}{3}$ for continuation. The values of the fuzzy certainty values are adjusted by iteratively applying all applicable rules to the contours comprising each hot spot. The rules are applied until all certainty values cease to change significantly. In

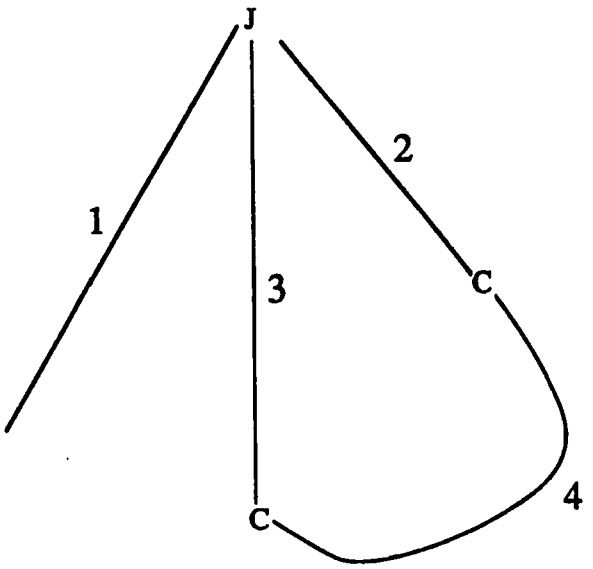


Figure 4.23. There is indirect evidence for grouping contour 1 with contour 4 via the join relation to contours 3 and 4 and though them to the continuation relations and then to contour 4. If the join and continuation relations are strong, then contours 1 and 4 should be grouped.

this respect, the approach can be thought of as rule-based relaxation.

In the current implementation, all of the rules are based on the relative locations of the endpoints of the contours, and the relationships of lines that are fitted to the ends of the contours. Line fitting is done using weighted least squares such that the edgels near the end of the contour are weighted somewhat more heavily than those further from the end. Examples of rules and more details of the algorithm can be found in Appendix A.

Figs. 4.24-4.26 and Table 4.2 show the results of applying the contour grouping algorithm to a complex scene. Fig. 4.24 shows an image of wooden blocks. Fig. 4.25 shows where the algorithm determined hot spots existed. Fig. 4.26 labels the contours with identification numbers for reference in Table 4.2. For a number of the hot spots appearing in Fig. 4.25, Table 4.2 gives certainty values for each type of relation that can exist between the contours. As can be seen, in many cases, the module assigns the relation that a person would have chosen based solely on local information. There are some cases where the module is not able to determine the correct relation between the

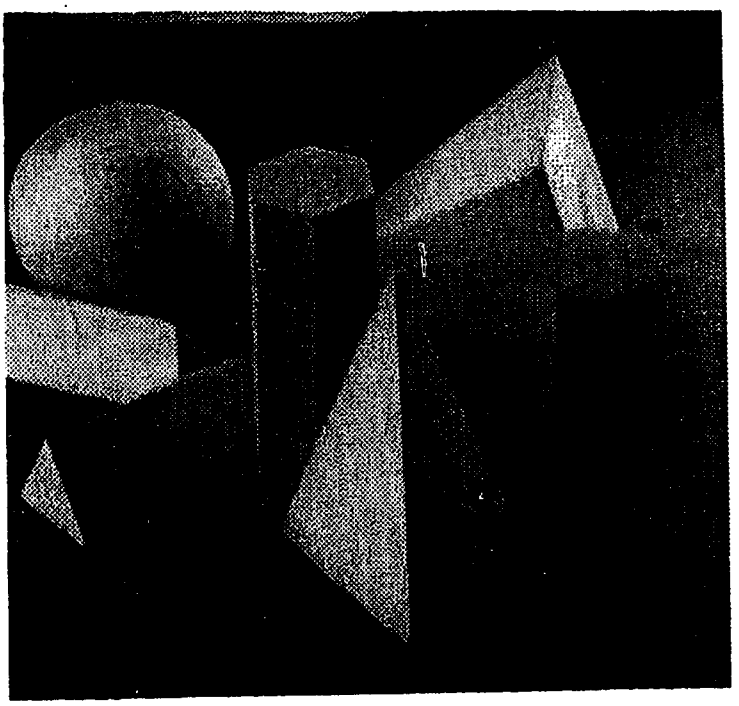


Figure 4.24. An image used to test the contour grouping module.

contours. However, such mistakes do not cause the overall recognition algorithm to fail, since the contour grouping information is only used to guide the formation of point-pair features that are likely to be meaningful. If these features do not lead Cyclops to good solutions, eventually it will use the less likely features. Thus, Cyclops will never fail to recognize an object due to the performance of the contour grouping module. Generally, however, the guidance provided by the contour grouping module helps.

Additional details of implementation of the contour grouping module are provided in Appendix A.

The other modules comprising Cyclops use the results of the contour grouping module in the form of the connection matrix, M. Once the relationships between contours have been determined, the elements of the connection matrix, M_{ij} can be found. If the contours i and j are directly related to each other by participating in the same hot spot, the value of M_{ij} could be set simply to the sum of the join and continuation labels. However, the

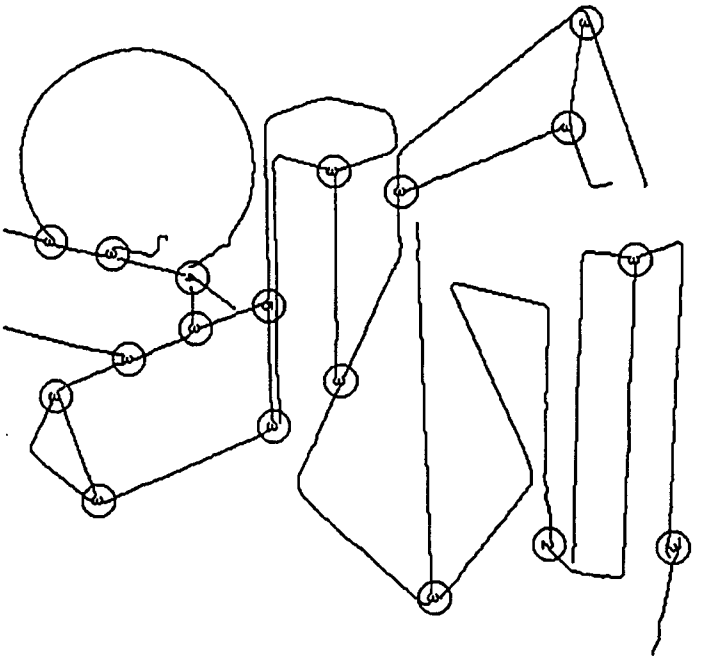


Figure 4.25. After edge detection and linking, the grouping module finds the hot spots, i.e., the places where contours may bear non-trivial local relations to each other.

evidence provided by the direct connection may be modified by indirect evidence from connections through other contours. Similarly, contours that are not directly connected may, nevertheless, be related via the indirect connections between them. How to best accomplish the computation of M using the information provided by the contour grouping is a topic for further research.

4.4 Multiview Models

We have discussed the mechanism for indexing representative model instances from multiview models using k-d trees. In addition, we have shown how to compute point-pair features that are invariant to image-plane translation, rotation, and scaling. We have also shown that in order to recognize objects in the absence of reliable object-attached features, it is necessary to use some type of multiview model. As described

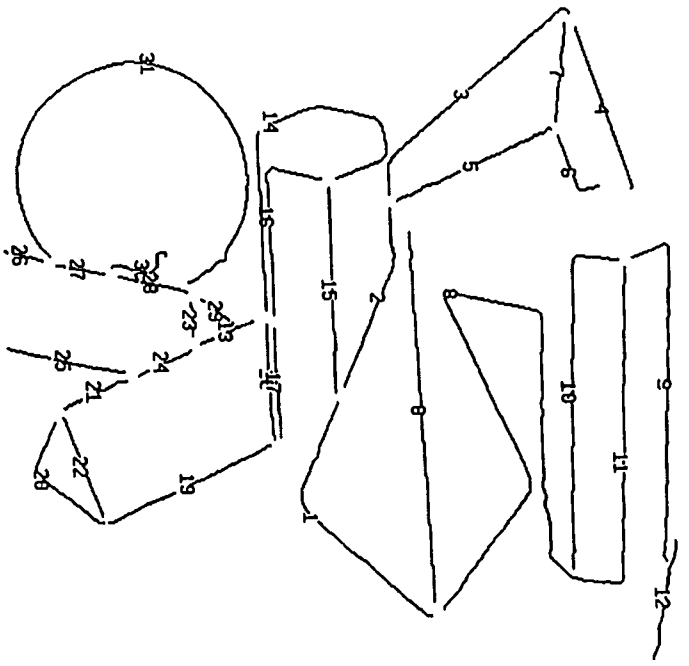


Figure 4.26. The ID numbers of the contours are shown above for use with Table 4.2

briefly in Chapter 2, multiview models in Cyclops consist of a dual representation: a *multiview feature representation* that contains the features and their attributes visible from a number of representative viewpoints, and a true *3-d surface representation* for use in the later stages of recognition, including attitude determination and incremental verification. Fig. 4.27 illustrates the multiview models employed by Cyclops.

The point-pair features we have described are used to index the models and the viewpoints in the k-d tree-based database. As we have seen, the hypothesis generation procedure uses this database to efficiently match features and generate hypotheses. However, we have not described how the viewpoints are chosen. In particular, how many of them should there be, and how should they be distributed on the surface of the viewing sphere. In answering these questions, we expect that there are many issues involved that will be the subject of further research. However, we will discuss some of the issues that we have considered in anticipation of implementing multiview models for Cyclops.

Contours	Contour	Continuation	Join	No-Connection
20	21	0.0	1.0	0.0
20	22	0.0	1.0	0.0
21	22	0.0	1.0	0.0
1	2	1.0	0.0	0.0
1	15	0.0	0.0	1.0
2	15	0.0	0.0	1.0
21	24	1.0	0.0	0.0
21	25	0.0	0.0	1.0
24	25	0.0	0.0	1.0
8	11	1.0	0.0	0.0
9	10	.21	.79	0.0
9	11	0.0	1.0	0.0
10	11	0.0	1.0	0.0
5	6	0.0	1.0	0.0
5	7	0.0	1.0	0.0
6	7	0.0	1.0	0.0
13	23	0.0	0.0	1.0
13	24	1.0	0.0	0.0
23	24	0.0	0.0	1.0
31	28	0.0	0.0	1.0
31	29	0.0	0.0	1.0
31	23	.33	.33	.33
28	29	1.0	0.0	0.0
28	23	0.0	0.0	1.0
29	23	0.0	0.0	1.0

Table 4.2. The table summarizes some of the results of running the contour grouping algorithm on the image shown in Fig. 4.24. The contour ID numbers are given in the first two coloumns, and the remaining columns contain the final values of the **Continuation**, **Join**, and **No-Connection** relations. Refer to Fig. 4.26 for the contour ID numbers.

The number and location of the representative viewpoints for the multiview feature representation of each model depends largely on the robustness of the attitude estimation module, or AEFMA, described in detail in Chapter 5, which is part of the overall

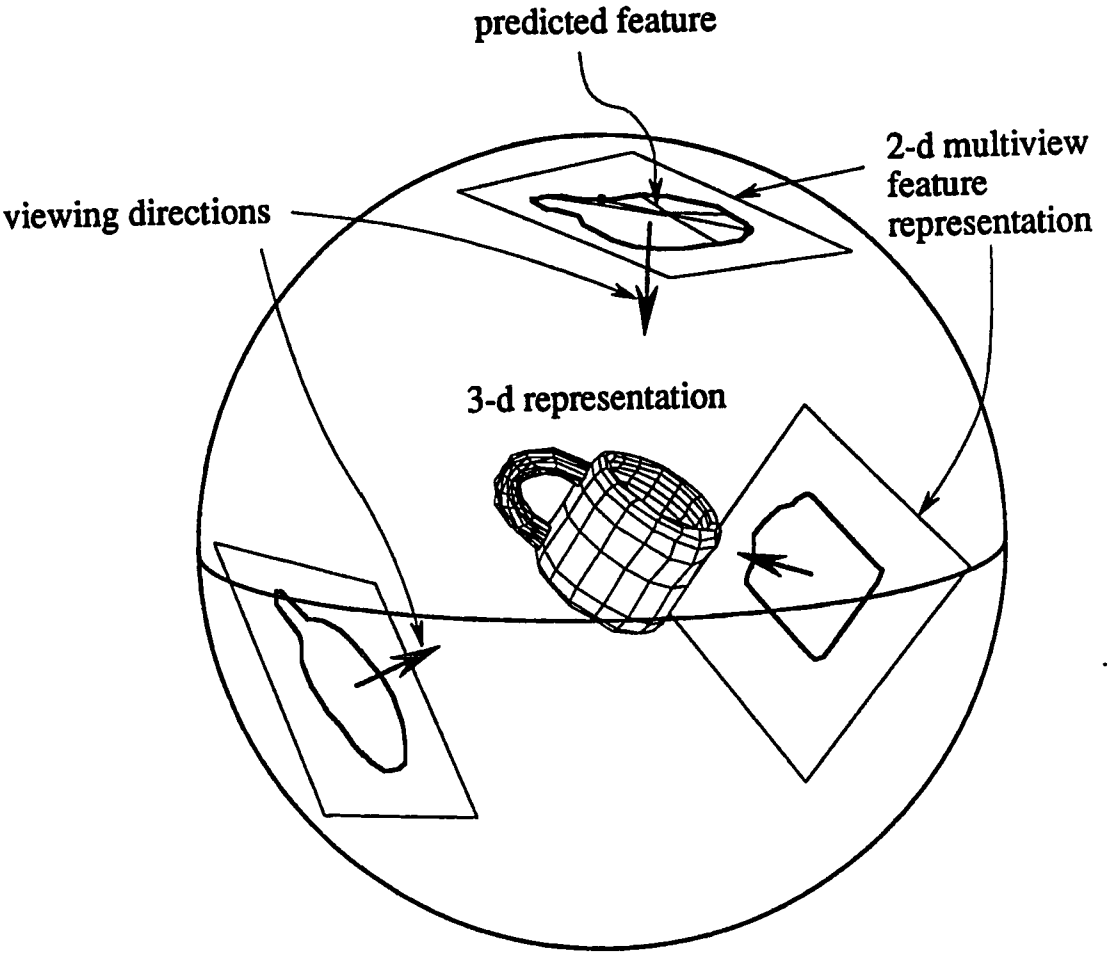


Figure 4.27. Shown is a multiview model of a coffee cup. The model consists of a 3-d surface patch representation and a 2-d multiview feature representation. The representative views, of which three examples are shown, each consist of the predicted 2-d edge contours as well as the predicted point-pair features that can be detected in these contours. Such a predicted feature is shown in the topmost representative view.

verification process. The attitude estimation module can determine the correct attitude of an object even though the initial estimate of the viewpoint, as provided by the hypothesis generator, is quite inaccurate, then few views need to be stored in the multiview feature representation of the multiview model. On the other hand, if precise estimation of the viewing parameters requires a fairly accurate initial estimate, then the coverage of the viewing sphere must be dense, and a large number of views may be required.

Fortunately, from most viewpoints, the attitude estimation module is able to correctly estimate the viewing parameters of an object even though the initial estimate of the viewing parameters is far from the correct values.

While AEFMA is often able to converge to the correct viewpoint from a distant viewpoint, error in the initial viewpoint that AEFMA is able to tolerate and still converge to the correct viewing parameters is a function of the initial viewpoint. In particular, viewpoints where the shape of the projection of the object changes rapidly tend to reduce the range of convergence of AEFMA. Thus, in such regions, the spacing between views in the multiview feature representation should be smaller, leading to denser coverage of the viewing sphere. Similarly, in regions where the shape of the projection of the object is changing smoothly with viewpoint, allowing AEFMA to converge to the correct viewing parameters from farther away, the representative views may be more widely spaced.

As will be seen in Chapter 5, the range of convergence of AEFMA is reduced when the object is occluded, eventually disappearing altogether. Thus, depending on the degree of occlusion that Cyclops is to tolerate, and the degree of accuracy required in the final viewing parameters, the covering of the viewing sphere by representative views may need to be considerably denser than the covering necessary for good performance on unobscured objects. Further, due to the difficulty of determining AEFMA's convergence in the presence of the many possible types of occlusion, uniform coverage may be the best solution. The density of coverage would be adjusted to attain the required level of performance.

Whatever the distribution of representative views over the viewing sphere is, the capability to compute a uniform covering of the viewing sphere using any number of points is likely to be useful. Existing methods for computing an approximately uniform covering of the viewing sphere usually start with a regular polygon, such as a dodecahedron or icosahedron [BB82], and subdivide the faces until a sufficiently dense covering is achieved. There are two drawbacks with these methods: first, the number of points

cannot be chosen arbitrarily; second, the covering is not as uniform as desired for some applications.

We have developed an iterative method that allows any number of points to be uniformly distributed over the surface of the viewing sphere. The algorithm models the points as repulsive charges confined to the surface of a sphere. Initially, the points are randomly distributed on the surface of the sphere. Each iteration of the algorithm computes the component of the net force tangent to the sphere acting on each point, and adjusts the location of the point in the direction of this component a distance proportional to its magnitude (as if the points were massless, but traveling through a viscous liquid). After the maximum tangential force on any of the points falls below a threshold, the algorithm stops. If the number of points is small, and is chosen to be equal to the vertices of a regular polygon, then the algorithm places the viewpoints on the vertices of the polygon, as shown in Fig. 4.28 for the case of a tetrahedron. However, any number of points may be used. For example, Fig. 4.29 show the result for 101 points.

4.5 Testing Cyclops' Hypothesis Generation Approach

Currently, Cyclops' hypothesis generation approach has been implemented in a 2-d recognition system. Two-dimensional recognition does not assume that objects are necessarily 2-d, as seen by the results demonstrating recognition of 3-d objects, parts of a switch assembly, in Section 4.5.6.5. Two-dimensional recognition assumes that the objects in the vocabulary, be they 2-d or 3-d, will be presented from a set of discrete viewpoints, each of which is treated as a separate 2-d model. This is precisely the approach taken by Cyclops through the use of multiview models to generate hypotheses for 3-d objects. Each view of a model, at this level, is treated by the system as though it were a separate object. Indeed, some of the "2-d" models in our 2-d recognition system are distinct, stable views of a single 3-d object. Thus, the 2-d system provides a reasonable test of Cyclops' hypothesis generation process.

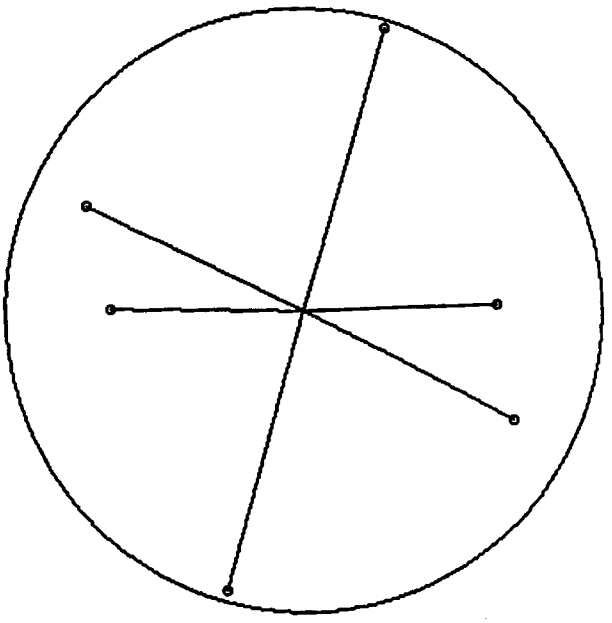


Figure 4.28. A result of running the algorithm for determining uniform spacing of viewpoints over the viewing sphere. In this case, six points were used. As can be seen, the resulting viewpoints fall on the vertices of an octahedron, as expected. However, unlike other methods for uniformly covering the sphere, any number of points may be used, as in Fig. 4.29.

The features used in the 2-d recognition experiment differ from the features described above for 3-d recognition. The 2-d features are based solely on the shape of edge contours in the locality of *single* critical points. For the purpose of 2-d recognition, we found such features to be sufficiently selective. Furthermore, the problem of determining perceptually significant features is greatly simplified since these features are not compound features.

The algorithm is a hypothesize-and-test approach, similar to the overall approach of Cyclops but with some important deletions. Most importantly, the 2-d algorithm has no way of estimating the 3-d pose of an object once a plausible hypothesis has been generated, such as AEFMA technique described in Chapter 5. In addition, the verification of method of the 2-d algorithm has a two level incremental approach. While this part

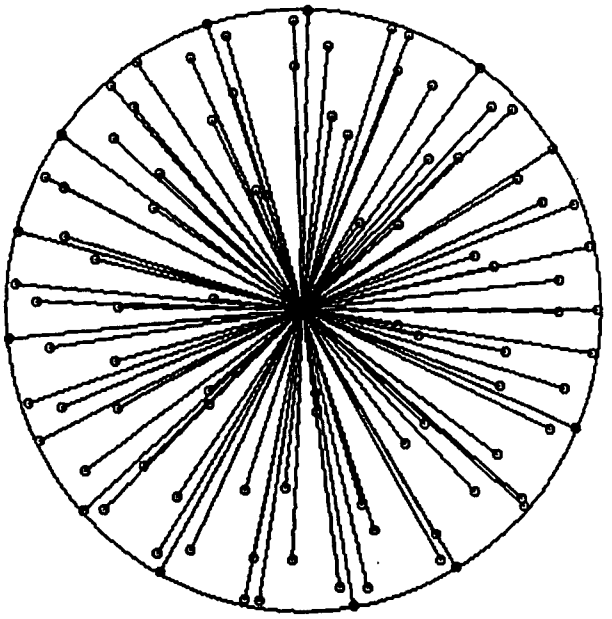


Figure 4.29. A result of running the algorithm for determining uniform spacing of viewpoints over the viewing sphere. In this case, 101 points were used. Other algorithms could not cover the sphere with 101 points. Further, even if they could, the covering would not be as uniform.

of Cyclops has not yet been implemented, the design calls for two additional levels in the incremental verification process. However, the principle of incremental verification is partially tested here.

In what follows, we will only briefly describe the hypothesis generation module of the 2-d algorithm since it is almost identical to that of CyclopsCyclops' hypothesis generation process has been described earlier in this chapter. The primary difference, as mentioned in the previous paragraph, is the features. Thus, we will spend more time describing the features. Parts of the feature computation are particularly interesting because of the Karhunen-Loève compression technique that enhances the efficiency of the hypothesis generation process. The verification technique will also be described here for continuity and referred to elsewhere when necessary.

4.5.1 Overview of the 2-d Recognition Algorithm

A problem which has received considerable attention in the computer vision literature is that of recognizing two-dimensional (2-d) partially visible objects in a gray scale image. In addition to being an important problem whose solution has many practical applications, it is an important step toward the solution of the more difficult problem of recognizing three-dimensional (3-d) partially visible objects in an image. The problem of recognizing partially visible objects is sometimes called the *bin of parts problem* because, in industry, parts are often presented for batch assembly piled in a bin. The general bin of parts problem (with no constraints on the objects that may appear in scenes except that they be rigid) has been described as the most difficult problem in automatic assembly [Mat76]. We describe a solution to the bin of parts problem where the objects are 2-d or have a small number of distinct viewpoints that may each be treated as 2-d objects, as in a multiview model.

This section discusses a 2-d partially visible object recognition algorithm that is a development of ideas first outlined in [GTM87]. In addition to being a novel approach to the problem of 2-d partially visible object recognition, we will also attempt to characterize the accuracy, robustness, and efficiency of the method as well as possible.

4.5.1.1 Related Previous Work

For a thorough review of the work relating to 2-d object recognition, the reader is referred to [CD86, Tur86, KJ87]. For reference, the following (in chronological order) are important references which, while not closely related to our work, also address the topic of 2-d object recognition: [Fu74, Pav77, BN78, Tro80, Bal81, Seg83, BS85, Bha84, CCL84, KK85, AF86, DG86].

The algorithm employs data-compressed vectors of samples from the slope–angle versus arclength (θ-s) representation of the edge contours of objects which are near to

high-curvature points of the contour (critical points). Other workers who have used the θ-s representation of edges are Perkins [Per77], Yam *et al.* [YMA80], Mckee *et al.*, [MA77], Turney [Tur86] and Tsui *et al.* [CT89]. Freeman [Fre77] has used critical points as features.

It is often true that algorithms that use selective features will be able to perform matching more quickly than those that do not. There are two reasons for this: first, there will be fewer highly selective features than unselective features, and second, selective features usually provide a large vector of attributes which can be used to quickly reject those <image-feature↔model-feature> pairings leading to a faulty hypothesis. These two properties usually allow such systems to reduce computation. A disadvantage of such systems is that, due to the scarcity of selective features, if the system is designed to handle overlapping or partially visible objects, then the degree of occlusion that can be tolerated by the system is reduced. Bolles and Cain [BC82] have used highly selective features to advantage. In their method, the features are comprised of a *focus feature* and a number of satellite features. The resulting composite features are very selective and rare, allowing Bolles and Caine to use an NP-complete matching procedure. Turney [Tur86] has also used highly selective features. His recognition procedure, however, remains robust to large degrees of occlusion in the images since it falls back on less selective features if the most selective ones are not present. While success of a recognition algorithm depends on the choice of good features, the design of the matching algorithm is even more critical. The matching process has been formulated as a subgraph matching problem [BC82, CCL84], as a tree search [Tro80, Goa83, AF86, GLP87], as a Hough Transform [TMV85, Tur86, KJ86, KSSS86], and a parsing problem [Fu74]. Our matching strategy does not fit neatly into any one of these categories, as it has elements of both tree searching and correlation over edge contours.

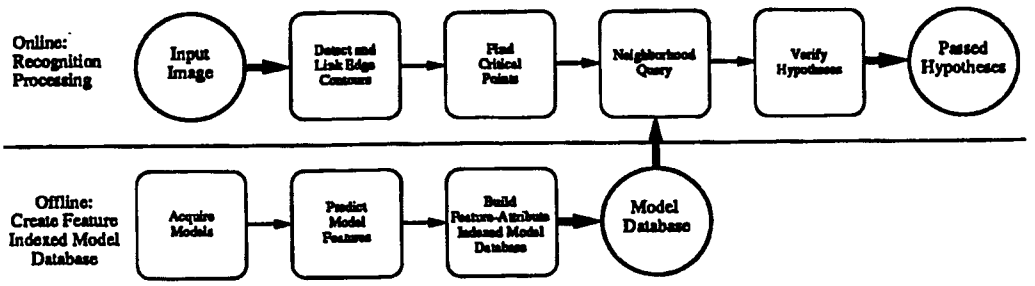


Figure 4.30. Steps in the 2-d recognition algorithm.

4.5.1.2 Considerations for the Design of a Highly Efficient 2-d Object Recognition Algorithm

Figure 4.30 shows the general strategy employed by our algorithm. Referring to the on-line half of Fig. 4.30, the input image is processed to extract edge contours. The edge contours are further processed to detect the features and compute their attributes. As in Cyclops' hypothesis generation process, the image feature vectors are then compared to all of the model feature vectors. Those model feature vectors that are close enough to an image feature vector, according to some metric, cause a hypothesis to be created. The hypothesis is associated with a viewing transformation that maps the matching model feature onto the image feature. These initial hypotheses must then be verified or rejected. The hypotheses that pass the final verification phase are those which are most likely to correctly predict whether an object appears, and, if so, where it appears.

The off-line preprocessing branch of Fig. 4.30 starts with the model generation phase. The 2-d models are typically generated by processing a set of training images, although they could by generated by CAD [TMV85]. Features are then extracted from the models in the recognition branch of the figure. in the same way as in the feature detection phase. The models are then stored in a k-d tree database so that the model features that match an image feature may be quickly retrieved. The advantages of this approach have been discussed earlier in this chapter.

Preindexing models by feature attributes has been done previously, although not to

the extent of Cyclops. For example, Turney [TMV86] sorts the set of model features by their saliency (recall that saliency formalizes the notion of selectiveness of 2-d features). However, the matching process is still a linear search. Knoll and Jain [KJ86] choose a set of features from the model so as to reduce overall recognition time. However, the improvement that this strategy can yield is limited since, again, the matching process is a linear search.

4.5.2 Feature Matching and Hypothesis Generation

The feature matching in the 2-d algorithm is identical to the matching described earlier for Cyclops. In particular, the models are indexed by the attributes of the features appearing in them and stored in a k-d tree database. This database is used to perform Euclidian $N_2(\mathbf{i}, \delta)$ queries, where $\mathbf{i}$ is an image feature vector, and delta determines the size of the neighborhood. The model features that are returned are used to create hypotheses, which are then verified by later stages of the algorithm.

4.5.3 Selection and Computation of Feature Vectors

While hypothesis generation is nearly identical under the 2-d algorithm and Cyclops, the features are different. The 2-d algorithm, like Cyclops recognizes objects entirely by the shape of contours. As in Cyclops the (x, y, θ)-s representation is used to represent the contours. However, the 2-d algorithm uses *only* critical points as features. In addition, the method of encoding the local shape of the contours near the critical points differs; the 2-d algorithm uses the Karhunen-Loève expansion applied to a vector of samples from the $\theta(s)$ part of the representation in the neighborhood of the critical points. We will refer to such features as critical point neighborhoods, or CPN's. For completeness, we now describe the computation of CPN's.

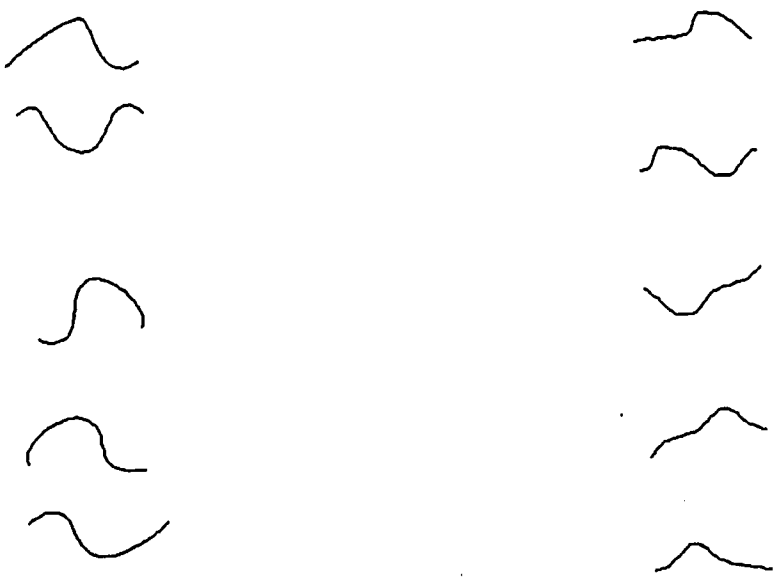


Figure 4.31. Above on the left are several typical puzzle piece CPN features in their Cartesian representation, and above on the right are the same features represented in θ-s space.

Critical point neighborhood features, while being much more selective on average than other segments of a set of contours, are nevertheless not a very efficient encoding of the important recognitive information. Examination of Fig. 4.31 reveals that CPN's are continuous and, in fact, rather similar in appearance. As we have noted, in practice the representation is discretized and contains a finite number of samples that comprise a vector. The similarities in Fig. 4.31 suggest that the elements of a CPN feature vector are highly correlated with each other, i.e., it is possible to predict quite accurately what the value of an element will be given the values of some other elements of the CPN feature vector. The Karhunen-Loève (K-L) expansion, a standard data compression technique in signal processing, takes advantage of highly correlated data. Rosenfeld and Kak [RK76] describe how the K-L expansion can be employed with success to compress picture data. We use it here to reduce the data needed to represent the CPN features described above.

Each sample of a CPN feature vector can be considered to be a component of a real vector of some dimension, say N. The set of CPN features can now be viewed as the result of trials of an underlying real random vector $\mathbf{X}$ of dimension N. We may assume that $\mathbf{X}$ is zero mean since, if it is not, the mean may be estimated and $\mathbf{X}$ adjusted accordingly. Let $\mathbf{R} = E[\mathbf{X}\mathbf{X}^t]$ be the auto-correlation matrix of $\mathbf{X}$. Since $\mathbf{R}$ is non-negative definite, there exists a set of orthonormal eigenvectors and associated eigenvalues of $\mathbf{R}$, ϕ_k and $\lambda_k \geq 0$ respectively, where $k = 1, \ldots, N$. Define the random variables $Y_k = \phi_k^t \mathbf{X}$. Without loss of generality, we may assume that the eigenvectors ϕ_k are ordered so that $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_N$. The K-L expansion says that $\mathbf{X}$ can be expanded in the following manner:

$$\mathbf{X} = \sum_{k=1}^{N} Y_k \phi_k, \tag{4.5}$$

where the Y_k are uncorrelated and $\mathrm{Var}(Y_k) = \lambda_k$.

Geometrically, the K-L expansion chooses a special basis in the N-dimensional vector space in which $\mathbf{X}$ is defined. This basis has the following property: the basis vector ϕ_1 defines the direction in which $\mathbf{X}$ has the greatest variance (i.e., Y_1, the projection of $\mathbf{X}$ in the ϕ_1 direction, has the maximum variance); the second basis vector ϕ_2 defines the direction in the subspace perpendicular to ϕ_1 in which $\mathbf{X}$ has the greatest variance, and so on until all dimensions are defined. Therefore, the K-L expansion chooses a basis which, when $\mathbf{X}$ is represented in terms of it, will concentrate the total variance of $\mathbf{X}$ into its lower numbered components. A subspace whose basis vectors ϕ_k are associated with those Y_k having small variance may be ignored with negligible effect upon the stochastic properties of $\mathbf{X}$. The result of this is that the original feature vectors can be projected onto a space of smaller (often considerably smaller) dimension and still retain most of their information. Fig. 4.32 gives an example of such a situation.

Before applying the K-L expansion to compress the data needed to represent the CPN feature, it is necessary to adjust the data to give it a zero mean to decorrelate the Y_k. Let $S = \{ \mathbf{f}_i, \ i = 1, \ldots, V \}$ be the set of all feature vectors in the object set

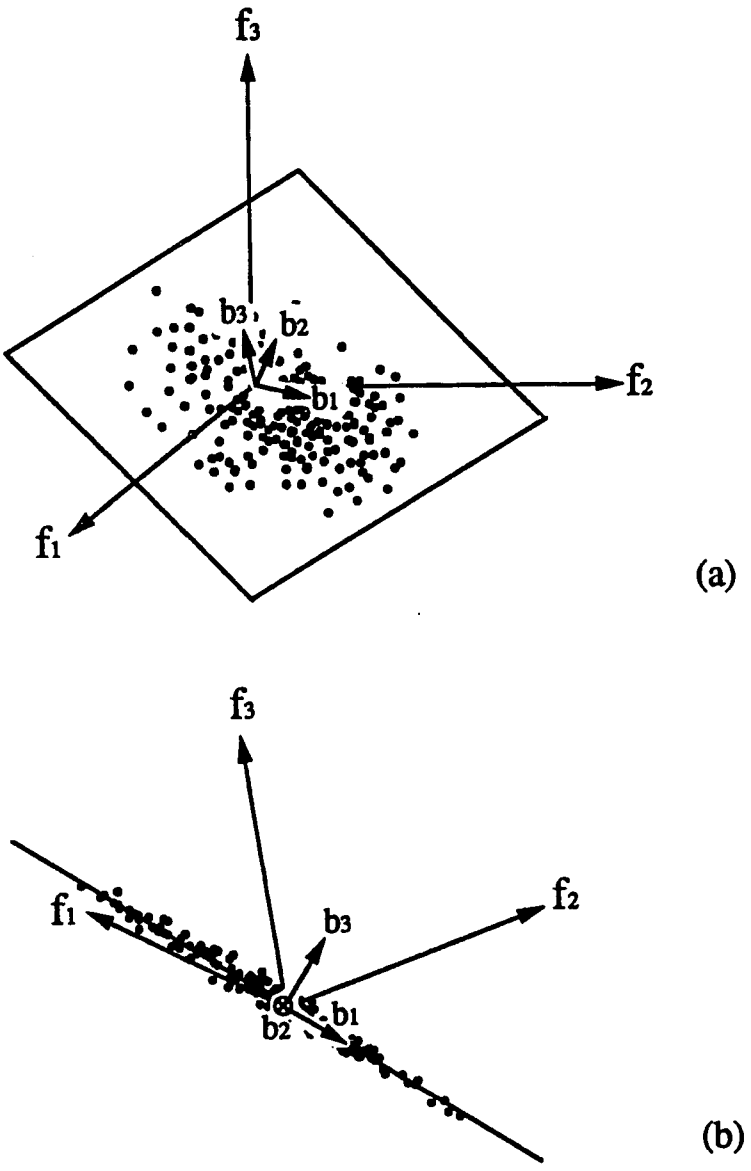


Figure 4.32. A hypothetical 3-d feature space is shown in (a) whose elements are highly correlated among themselves (dots represent features in the space). The correlation can be seen from the degree to which the features cluster near the plane. Shown in (b) is a different view of (a) looking with $\mathbf{b}_2$ going into the paper. The vectors $\mathbf{b}_1$ and $\mathbf{b}_2$ span the plane while $\mathbf{b}_3$ is orthogonal to it. When represented in terms of the basis $\mathbf{b}_1$, $\mathbf{b}_2$, and $\mathbf{b}_3$, the $\mathbf{b}_3$ component of the vectors will be near zero and may be ignored, achieving a degree of data reduction. The K-L expansion allows such a basis to be computed.

(extracted from models or training images), where V is the number of CPN features found in the training images. The set of zero mean feature vectors derived from S is $T = \{\mathbf{f}_i = \mathbf{f}'_i - \mathbf{m},\ i = 1, \ldots, V\}$, where $\mathbf{m}$ is the sample mean of S. The autocorrelation matrix, $\mathbf{R}$, can then be estimated from T by the formula

$$\mathbf{R} = 1/V \sum_{k=1}^{V} \mathbf{f}_k \mathbf{f}_k^t. \tag{4.6}$$

Using this estimate of $\mathbf{R}$, the values of ϕ_k and λ_k can be computed, and then the the K-L expansion formula can then be applied. The data reduction is effected by retaining only those basis vectors ϕ_k associated with the Y_k having the largest variances and then projecting the original features onto the subspace spanned by the retained ϕ_k. Define the total variance of $\mathbf{X}$, σ_x^2, as $E[\mathbf{X}^t\mathbf{X}]$, then $\sigma_x^2 = \sum_{k=1}^{N} \lambda_k$, since the ϕ_k are an orthonormal set. The number of ϕ_k's retained, L, is determined by the fraction of σ_x^2 we wish to retain[1]. The reduced feature vector, $\mathbf{r}$, of any CPN feature $\mathbf{f}$ can then be computed by the formula $\mathbf{r} = \mathbf{Pf}$, where $\mathbf{P}$ is the $L \times N$ matrix whose rows are the L retained ϕ_k ordered such that the ϕ_k with the largest associated variance appears at the top, the ϕ_k with the second largest variance appears second from the top, and so on to the bottom where the last retained ϕ_k (associated with the smallest variance) appears.

Fig. 4.33 shows the K-L basis vectors, ϕ_k, which have been computed using all of the CPN features from the set of models of jigsaw puzzle pieces, the associated variances of the Y_k, and the reduced feature vectors. In our case, we required that 98% of the total variance be retained; only 5 dimensions out of total of 45 were necessary to achieve the 98% variance figure. This is a reduction in data by nearly an order of magnitude. Finally, Fig. 4.34 shows the projections of some typical CPN features onto the reduced

---

1 The fraction is a design parameter. Adjusting the fraction allows data reduction to be traded off for informativeness. If the fraction is very close to 100%, there will be less data reduction and the reduced features will represent the original features more closely. On the other hand, a lower fraction will increase the data reduction at the expense of a less perfect representation of the original features. As discussed in the text, a fraction quite close to 100% (e.g. 98%) still yields a large data reduction.

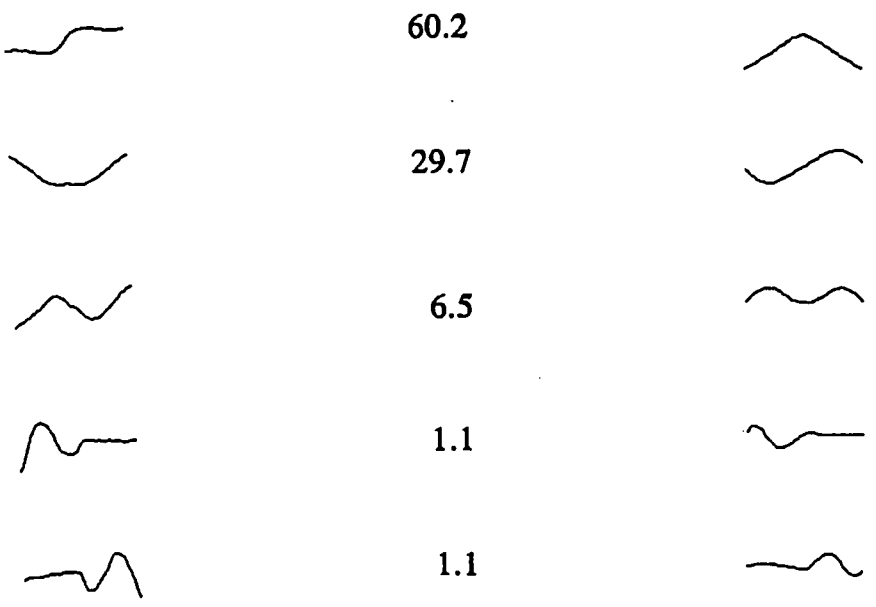


Figure 4.33. Above on the left are the basis vectors represented in θ-s space. They result from applying the K-L expansion to all of the CPN features in the model set (in this case, a set of ten puzzle pieces) In the middle of the figure is the percentage of the total variance associated with each basis vector. At the far right are the Cartesian representations of the basis vectors.

basis obtained from applying the K-L expansion to the CPN's of a set of puzzle piece contours.

We have previously alluded to an important synergy that exists between the matching method and the feature computation technique. The k-d tree matching would be significantly less efficient if the K-L expansion were not applied to the original feature vectors to yield the reduced feature vectors with uncorrelated, high-variance components. As discussed earlier, the logarithmic complexity of retrieval from a k-d tree is only attained if no more than a small proportion of the nodes that are visited require both subtrees to be examined. If this requirement cannot be met, then the performance of a retrieval degrades to its worst case linear complexity. If the data is highly correlated, and a k-d

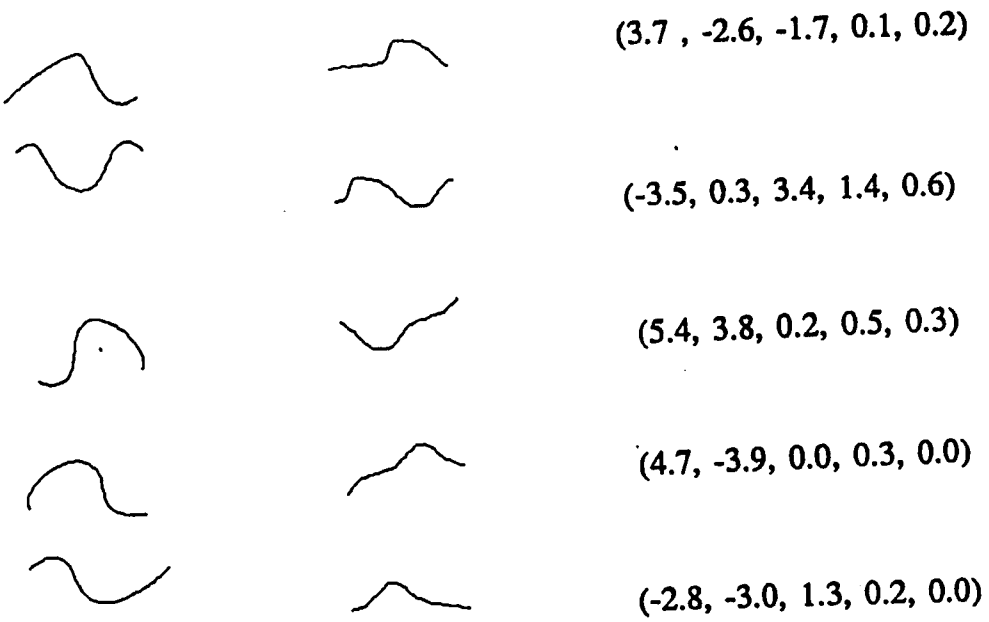


Figure 4.34. On the left are shown some typical puzzle piece CPN features in Cartesian space. In the middle are their θ-s representations. On the right are their projections onto the reduced basis made up of the five vectors shown in Fig. 4.33.

tree is built, it is much more likely that both subtrees will need to be searched during a retrieval than if uncorrelated, high variance data is used. Therefore, it is advantageous for the overall performance of the k-d tree matching that it be use in conjunction with feature vectors that have been reduced using the K-L technique.

4.5.4 Hypothesis Verification

We now turn to the problem of hypothesis verification. In our framework, a hypothesis is a prediction about what may appear in the image, and verification is the process of comparing the prediction with some set of observations. Hypothesis verification consists of a detailed comparison of the model of the hypothesized object at the hypothesized pose with the image (or data derived from the image). The purpose of the comparison is to

gather evidence, positive or negative, about the hypothesis in question. The result of the comparison is a score which rates the strength of the hypothesis. The score depends in some way upon how closely what actually appears in the image matches what is predicted to appear. All the available hypotheses are rated, and those that have enough support are kept. These remaining hypotheses are the algorithm's best guesses as to which objects appear in the image, and their poses.

In terms of the discussion in Sec. 4.5.2, a hypothesis is a model that has a reduced CPN that falls within the δ-neighborhood of some image feature identified during the feature detection stage. The model, then, represents our guess at the specific object in the image that gave rise to the image feature.

The model of the object contains the reduced feature and its position within the model. More specifically, the model of the object is the set of all of its (x, y, θ)-s contours together with a list of the poses of all CPN features in the model. This information allows the model's pose to be transformed to the pose indicated by the feature found in the image (and therefore allows the list of critical points from the model to be transformed to their expected locations in the image). It will be convenient in the rest of the discussion to speak of a hypothesis as being the contours and critical points of the model after the pose transformation which aligns the model feature to the image feature (where the model feature matched the image feature), not just the information necessary to effect the transformation. Fig. 4.35 illustrates the process of transforming the model's contours and critical points so that the matching model and the image features are brought into alignment.

Our verification scheme bases its decision about whether to pass or fail a hypothesis upon two statistics: a fraction of critical points matched Q_{cp}, and a fraction of boundary matched Q_b. We chose these two statistics since past experience showed them to be effective decision variables for a wide variety of objects. In addition, other researchers have used similar measures successfully. For example, [BC82] describe a measure that

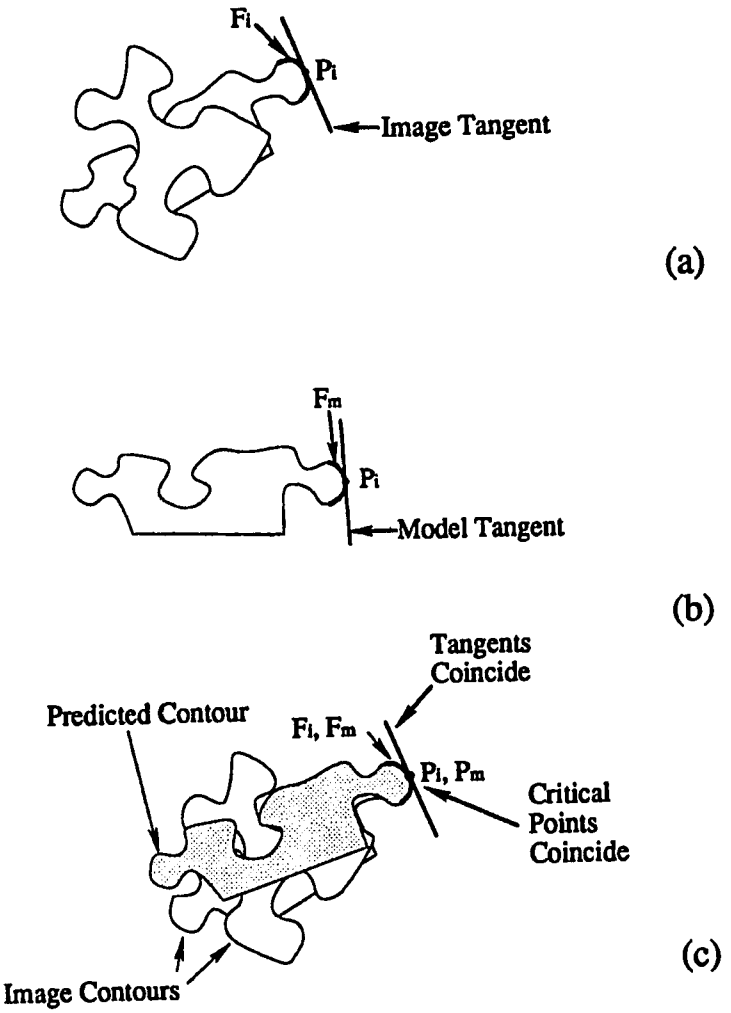


Figure 4.35. This figure illustrates how a model is transformed into a hypothesis using a feature that was matched during the feature matching stage of the algorithm. The image is shown at (a), the model at (b), and the image with the dotted outline of the hypothesis superimposed in (c). The reduced model CPN feature vector derived from $\mathbf{f}_m$, shown in (b), matches the reduced image CPN feature vector derived from $\mathbf{f}_i$, shown in (a). The critical points associated with these features are P_m and P_i respectively. To create a hypothesis, P_m is translated so that it coincides with P_i and is rotated so that the tangent to the model contour at P_m is aligned with the tangent to the image contour at P_i. (c) shows the gray-filled hypothesis resulting from this process.

is similar to or computation of Q_b, and [AF86] describe a measure that is similar to Q_{cp} except that they use line segment features instead of critical points, as in this work.

Unlike other researchers, we have chosen to use two measures as we have found that each possesses distinct strengths. In particular, Q_{cp} is effective for objects possessing many critical points; it heavily weights the most informative portions (the critical point neighborhoods) of an object's contour. However, the statistic Q_{cp} alone is not adequate for all kinds of objects. Some objects, such as nails, have relatively few critical points. In these cases employing Q_b in conjunction with Q_{cp} is appropriate since these objects often have only three or four critical points visible; all the rest may be occluded by other objects. This is generally more than adequate for generating hypotheses, but it is inadvisable to rely solely on properties of the critical points for hypothesis verification. For example, Q_{cp} becomes very sensitive to accidental matches when there are few critical points on the object. On objects with few critical points, all contour segments are roughly equally informative, hence the presence of any hypothesized contour in the image can be regarded as evidence that the hypothesized object was indeed present in the scene.

The idea of searching for features predicted by the model, and using their presence to strengthen hypotheses is not new. As mentioned above, a quality measure very similar to Q_{cp}, with the difference that the features were line segments rather than critical points, can be found in [AF86]. In that work, however, the process is taken a step further by using a Kalman filter to recursively update a least-squares estimate of the model's position and orientation each time a new image line segment is found that matches closely to a model line segment. This step provides a more precise positional estimate than our method. It would be a simple matter to add this capability, if the application requires precise pose information. A simple least-squares fit would be sufficient, especially in light of the work in [Tur86] where it is shown that the Kalman filter reduces to a least-squares fit when it is used for estimating the pose of motionless objects.

We now detail the computation of the statistics. The value of Q_{cp} is computed by searching a spatial neighborhood of each critical point from the hypothesis for a matching critical point in the image, and incrementing a count if one is found. In order to match, a critical point from the image must possess roughly the same orientation and the same sign of curvature as the hypothesized critical point against which it is being matched. The orientation of a critical point is simply the contour's θ value at the critical point (recall that we consider the pose transformation to have already taken place). The value of Q_{cp} is then given by

$$Q_{cp} = I_{cp}/M_{cp}, \tag{4.7}$$

where I_{cp} is the number of critical points in the hypothesis which were found to match an image critical point, and M_{cp} is the total number of critical points in the model. In order to check quickly whether there are any critical points in the image matching a hypothesized critical point, a second k-d tree is employed. For reasons that will shortly become apparent, we shall call this k-d tree the *pose tree*. The pose tree is built during the feature detection stage when the CPNs are being found. The k^{th} CPN in the image is assigned a key vector $(x_k, y_k, \sin\theta_k, \cos\theta_k)$, which we shall call the pose vector. The elements of the pose vector are defined as follows: the ordered pair (x_k, y_k) is the coordinates of the k^{th} critical point in the image, and θ_k is the slope angle of the contour at the critical point. The sine and cosine of θ_k were used in place of θ_k itself to avoid branch discontinuity problems associated with direct representation of slope. Each hypothesized critical point is also assigned a pose vector. The pose tree is then used to perform a range query to retrieve all image critical points with roughly the same pose as the hypothesized critical point. Let the hypothesized critical point have the pose vector $(x_h, y_h, \sin\theta_h, \cos\theta_h)$. The range is then defined by the 4-dimensional interval $(x_h \pm dx, y_h \pm dy, \sin\theta_h \pm d\sin\theta, \cos\theta_h \pm d\cos\theta)$. The values of dx, dy, $d\cos\theta$, and $d\sin\theta$ are not critical; they must be fairly large to allow for slight differences in pose of the hypothesis and an instance of the object in the image. We used 4 pixels for dx and

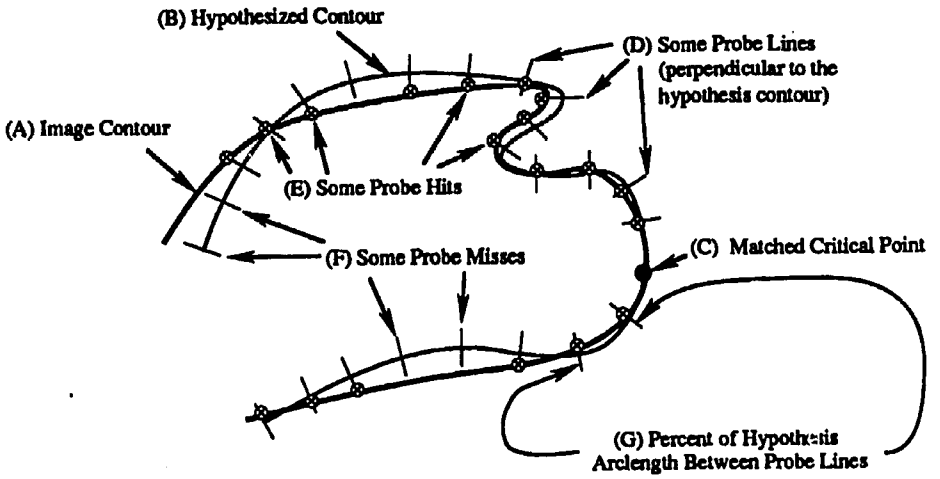


Figure 4.36. Here we illustrate the calculation of the fraction of boundary matched statistic, Q_b. Shown is an image contour (A) and a corresponding hypothesized contour (B). The hypothesis contour contains a CPN feature which matched an image CPN at (C). As explained in Fig. 4.35, the pose of the hypothesized part has been transformed so that the matched features have the same pose. In order to calculate the fraction of boundary matched, probes (D) are made perpendicular to the hypothesis contour in lines extending five pixels on either side of it. If a probe line intersects an image contour, a probe hit (E) has occurred, otherwise it is a miss (F). For each probe hit that occurs, Q_b (which is initialized to 0) is incremented by the fraction of the total hypothesis contour between two consecutive probe lines (G).

dy, and 0.5 for $d\sin\theta$ and $d\cos\theta$. If the list returned by the range query is not empty then the count I_{cp} is incremented. This process continues until all of the hypothesized critical points are checked, at which time Q_{cp} may be computed.

We have discussed the computation of the first statistic which is based on the count of matched critical points. We now explain how we perform the computation of the second statistic, the fraction of boundary matched. The mechanics of performing the boundary comparison are quite straightforward. Refer to Fig. 4.36 for an illustration of the process. The image contours are first drawn onto a bitmap to allow easy checking of the spatial proximity of contours (the contours are drawn white on black). Following the step of drawing the image contours, the contours of the hypothesis are traced, sample by sample,

creating the Cartesian representation of the hypothesis. As before, the transformation from the model to the hypothesis is chosen so that the pose of the hypothesized CPN feature is at the same pose as the image CPN feature that it matched. At fixed intervals of arclength, ds, a probe is made along a line perpendicular to the hypothesis contour. The pixels on the probe line are generated by Bressenham's line algorithm [Bre65] such that the line probed is perpendicular to the hypothesis contour, and the pixels in the line are checked up to five pixels on either side of the contour. If a white pixel (which corresponds to a point on one of the image contours) is encountered in the probe, the fraction of boundary statistic, Q_b, is incremented by the quantity ds/s_t, where s_t is the total arclength of all the contours making up the hypothesis. The process continues until all of the contours making up the hypothesis have been probed.

We have discussed the methods used to compute Q_{cp} and Q_b, but we have yet to explain how these statistics are used to make the decision to accept or reject a hypothesis. As would be expected, the optimal decision regions depend upon the object set as well as the degree of occlusion allowed. While we do not find optimal decision regions, we nevertheless desire to be general enough to get good performance for a wide variety of object sets and degrees of occlusion. In particular, a hypothesis is accepted if the ordered pair (Q_{cp}, Q_b) falls into the following region:

$$\{(x,y):\ x > T_1,\ y > T_2,\ \text{and}\ \beta x + (1-\beta)y > T_3\}\,. \tag{4.8}$$

Otherwise, the hypothesis is rejected. The three thresholds and β are chosen to give the best performance with the given object set. Fig. 4.37 shows a typical acceptance region.

4.5.5 Summary of the Algorithm and Complexity Analysis

The previous three sections have dealt in detail with the matching, feature detection, and hypothesis verification. In this section, we shall summarize the algorithm and analyze its complexity.

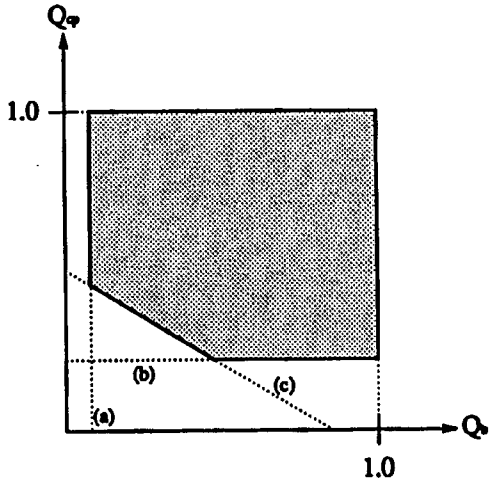


Figure 4.37. In order to be accepted, a hypothesis must fall into the gray region in the above diagram. This region is the intersection of the half-planes defined by vertical line (a), horizontal line (b), and oblique line (c), and bounded above by 1.0 in both the Q_{cp} and Q_b directions. Four parameters determine the shape of the region; they are described in the text. Optimal values of the parameters vary with the set of objects and must be determined experimentally. Performance is not sensitive to the precise values of the parameters.

Let N be the number of model features, let b be the average number of critical points per unit arclength of contour in the model set, let P be the number of objects in the image, and let I be the number of features detected in the image. We shall examine the off-line computation first, ignoring the low level operations of edge detection and edge linking. The off-line computation is composed of the following series of steps (which have been discussed in detail above): critical point detection, neighborhood extraction, K-L expansion, basis reduction, projection of model CPN's, and building of the k-d tree. These steps have complexity $O(N)$, $O(N)$, $O(N)$, $O(1)$, $O(N)$, and $O(N \log N)$ respectively. Thus the entire procedure has complexity $O(N \log N)$.

The on-line portion of the algorithm is composed of two major parts: first a sequence of steps that are executed only once for each image that the algorithm is asked to process, and a second sequence of operations that form a loop. We have written the on-line portion of the algorithm in Pascal-like pseudocode shown in Fig. 4.38.

```
Procedure ONLINE
BEGIN
      DETECT_CRITICAL_POINTS;
      EXTRACT_NEIGHBORHOODS;
      POSE_TREE_CONSTRUCTION;
      PROJECT_CPNS;

LOOP:
      GET_NEXT_FEATURE;
      NEIGHBORHOOD_SEARCH;
      VERIFY_HYPOTHESES;
      REMOVE_ASSIGNED_FEATURES;
      IF MORE_FEATURES THEN LOOP ELSE DONE END
```

Figure 4.38. The pseudocode for the recognition algorithm.

The function of the first four steps should be clear from their names and the earlier discussion. The procedure DETECT_CRITICAL_POINTS has complexity $O(I/b) = O(I)$; the procedure EXTRACT_NEIGHBORHOODS has complexity $O(I)$; the procedure POSE_TREE_CONSTRUCTION has complexity $O(I \log I)$; and the procedure PROJECT_CPNS has complexity $O(I)$. Thus, the total complexity of the four steps prior to the loop is $O(I \log I)$. We now examine the loop body. The procedure GET_NEXT_FEATURE retrieves the next available feature from the set of features remaining to be processed. This procedure has complexity $O(1)$. The next step in the loop is NEIGHBORHOOD_SEARCH. Recall that this procedure retrieves all image features within a neighborhood of the image feature that was obtained by GET_NEXT_FEATURE. This procedure has average complexity $O(\log N)$. Following NEIGHBORHOOD_SEARCH is VERIFY_HYPOTHESES. As described in Sec. 4.5.4, this procedure decides whether the hypotheses generated by NEIGHBORHOOD_SEARCH are good enough to be considered final hypotheses. This procedure is complexity $O(\log I)$ since it queries the pose tree. Next, the procedure REMOVE_ASSIGNED_FEATURES removes from the set of image features remaining to be processed those features that have been determined by the hypothesis verification stage to belong to a final hypothesis. This is an $O(1)$ step. Finally, a test based on

MORE_FEATURES is made. MORE_FEATURES returns TRUE if there are additional image features to be processed by the algorithm and FALSE otherwise. This is also an $O(1)$ step. Thus, the complexity of the loop body is $O(\log I + \log N)$. The loop will be executed at most I times (usually considerably fewer than I times). This leads to a combined complexity of $O(I \log I + I \log N)$ for the loop. Combining this with the complexity of the previous four steps yields $O(I \log I + I \log N)$ as the complexity of the entire on-line recognition procedure.

From an information theoretic point of view, the logarithmic dependence of the complexity on the number of model features is a lower bound. This behavior gives us a great deal of latitude to add as many redundant models as necessary of each object in order to achieve good performance without significantly degrading the speed of the algorithm. As will be described in the next section, redundant models help to combat widely varying lighting conditions, as well as perspective effects of objects that are really 3-d but which the system is treating a 2-d.

4.5.6 Experimental Results

This section gives results of experiments with the 2-d recognition algorithm that we have described. We first describe the methods we used in the experiments.

4.5.6.1 Methods

We now present some of the experimental results which we have collected from running our recognition algorithm on a number of images. All images were obtained from a CCD camera at 256×256 resolution. Each pixel was digitized to 8 bits. We then preprocessed all images by applying a Canny edge detector [Can83], and we then applied a simple linking algorithm to trace the contours. After the linking step, we resampled the contours so that both the Cartesian and the θ-s contours were sampled at uniform

intervals of arclength. To accomplish this, we used an operation developed in [Tur86] which simultaneously smoothes the contours, resamples them, and generates both the resampled Cartesian and θ-s contours of the image.

We ran the algorithm on two sets of objects: a set of ten puzzle pieces and a set of ten switch parts[2]. Each image contained all the objects. The puzzle pieces are a set of truly 2-d objects which provide a good benchmark. The switch parts are not true 2-d parts. However, by treating each view of a distinct stable position as a 2-d object, we were able to use our 2-d algorithm to recognize the 3-d switch parts.

We painted the jigsaw puzzle pieces white to cover the pictures normally appearing on jigsaw puzzles. In all experiments, the parts were scattered randomly in a tray, with some effort made to encourage occlusions. The switch assembly was not painted, and was comprised of a number of specular metallic and non-specular plastic parts. The lighting used was a simple fluorescent desk lamp with a movable arm. This was used to reposition the light prior to the acquisition of each image, allowing us to measure robustness to lighting changes.

The algorithm was run on an Apollo series DN570 color workstation, a 5 MIPS machine (roughly).

4.5.6.2 Off-line Preprocessing

The off-line processing needs to be performed only once for each distinct set of objects on which the system is required to work. After a training image was preprocessed as described above, the next step was the assignment of contours to object labels (our training images typically contain several objects). The CPN features of the model contours were then extracted using a simple one-dimensional derivative of a Gaussian edge detector as described in Sec. 4.5.3. We then applied the K-L expansion to the CPN features to obtain

---

[2] These parts were provided by the Air Force. They are some of the same parts that Cowan *et al.* [CCL84] used to test ACRONYM.

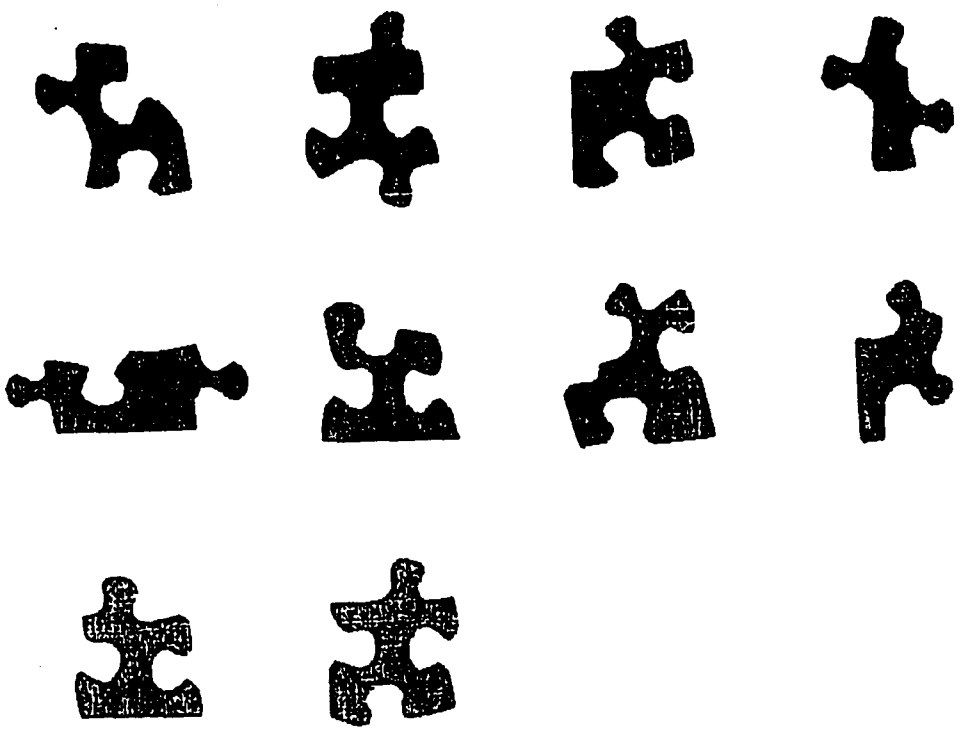


Figure 4.39. Models of the Jigsaw Puzzle Pieces.

the reduced basis, and the model CPN's were then projected onto the subspace spanned by the reduced basis, also per Sec. 4.5.3. The projections of the CPN's were stored for use by the on-line recognition procedure.

Fig. 4.39 shows the set of ten puzzle pieces which form our first model set. Similarly, Fig. 4.40 shows the set of views the algorithm will use to form the model set for the switch parts. Note that in Fig. 4.40, some of the models are of the same stable position of a part. It was advantageous to employ these redundant models for a number of reasons. First, the fact that the switch parts are really 3-d implies that their aspects change slightly depending upon where in the field of view they are located. Secondly, some of the switch parts were metallic and quite reflective. We found that for these parts, lighting changes could alter the shape of some of the critical points enough that the algorithm could not recognize a part (this is especially true of the reflective parts that have few critical points). Adding a model whose training image was taken in different lighting

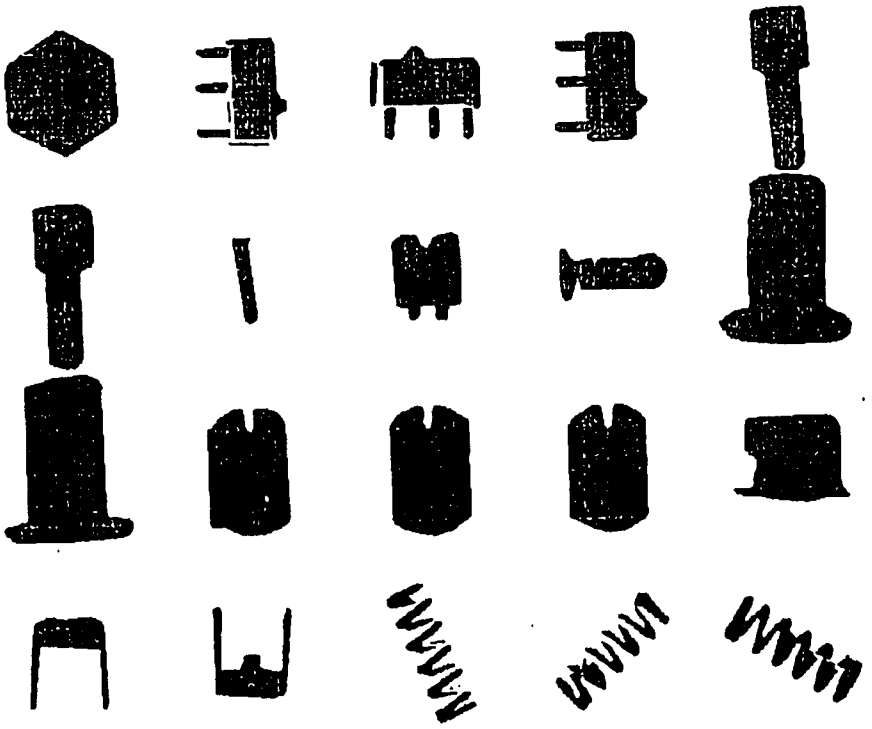


Figure 4.40. Models of the Switch Parts. The microswitch models are shown in the second, third and fourth positions from the left in the top row.

greatly improved the robustness of our algorithm with respect to lighting changes for such parts. Finally, and again related to lighting changes, one part (the microswitch—see Fig. 4.40) had details that were near the limit of our resolution and were extremely sensitive to lighting: images taken in one set of conditions would show the detail while images taken in other conditions would not show the detail. To solve the problem, we included one model for each case. Because our matching algorithm is only logarithmic in the number of model features, the addition of more views did not significantly affect the runtime of the algorithm. In fact, doubling the number of views would increase the matching by just one step.

False Alarm Statistics				
Object Set	# with 0 F.A.s	# with 1 F.A.	# with 2 F.A.s	Total # of Images
Puzzle Parts	6	0	0	6
Switch Parts	9	4	1	14

Table 4.3. False alarm (F.A.) statistics for experiments on the two sets of objects. The rightmost column contains the total number of images, and these are divided into the number of them which had zero, one, or two false alarms in the first, second and third columns. None of the images had more than two false alarms. As can be seen in the table, none of the jigsaw puzzle images had any false alarms.

4.5.6.3 Recognition Processing and Results

In this section we experimentally assess the accuracy, robustness, and efficiency of our algorithm. We give three means of characterizing the accuracy of our algorithm.

1. Plots of the percentage, p, of the objects which the algorithm recognized correctly versus the percentage of the object's contour which is exposed (Figs. 4.42 and 4.44). Note that the percentage of objects missed (i.e., not found by the algorithm when they should have been found) is $100 - p$.

2. The number of false alarms generated in each image (Table 4.3). This is the number of times that the algorithm predicts an object to be in the image which is not actually there.

3. Images before the recognition procedure is run, and the same images with the final hypotheses superimposed. This yields a reasonable, although qualitative, measure of the positional accuracy of our algorithm.

To characterize the robustness of our algorithm, we rely on the fact that we have run a sizable number of experiments under widely differing conditions, namely, two object sets, several lighting setups, and many object placements. In addition, the plot of the

percentage of objects recognized correctly versus the percentage of the object's contour visible makes explicit the algorithm's robustness with respect to occlusion. Finally, to characterize the algorithm's efficiency, we give (in addition to the complexity) average runtime of the algorithm from the feature detection step onward for each object set.

After the preprocessing described earlier in this section, the on-line portion of our algorithm proceeds as described in Sec. 4.5.5.

4.5.6.4 Puzzle Pieces

Fig. 4.41 shows an example result of running our algorithm on an image of overlapping puzzle pieces. Additional results may be found in Appendix B. Fig. 4.42 illustrates the algorithm's robustness with respect to occlusion. It shows a plot of the percentage of puzzle pieces correctly recognized versus the percentage of the boundary of the puzzle pieces exposed. As can be seen from the figure, none of the puzzle pieces is wrongly classified. Also, any puzzle piece with more than 55% of its boundary exposed is recognized correctly. For those pieces with less than 55% of their boundaries exposed, the algorithm sometimes has no hypothesis good enough to consider as a final hypothesis. However, the false alarm numbers in Table 4.3 show that if the algorithm does have a final hypothesis, it will be correct with near certainty.

The four variables which determine the decision region specified by (4.8) for the set of puzzle parts are as follows: $\beta = 0$, $T_1 = .3$, $T_2 = 0$, and $T_3 = 0$. In other words, for this part set, only Q_{cp} is used to decide whether or not to keep a hypothesis. The reason for this is simply that the puzzle pieces have many critical points and, as discussed in Sec. 4.5.4, for such objects, Q_{cp} is really the only decision variable needed.

The average time to finish an entire image from the stage of feature detection onward was 2.0 seconds for test images containing ten puzzle pieces each. Humans who are familiar with the puzzle pieces generally took at least 20 seconds to recognize as many puzzle pieces as they could from an image. They could usually recognize more pieces

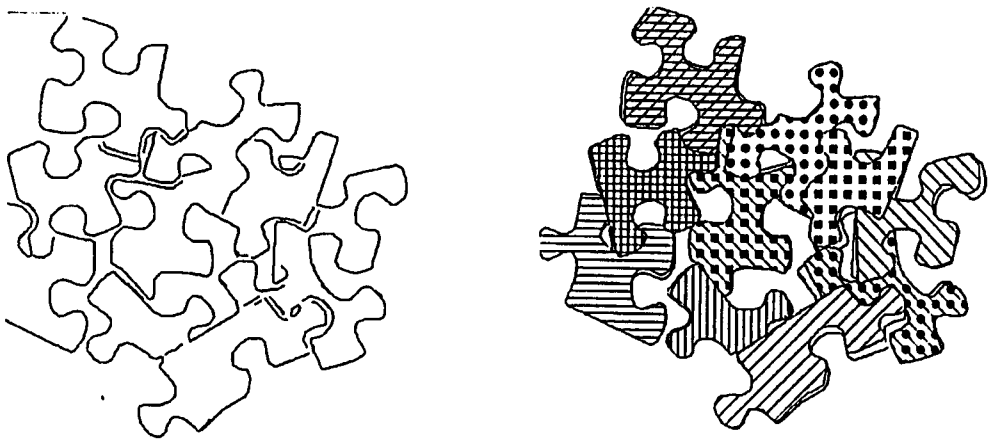


Figure 4.41. Above is a representative example of running the algorithm on an image of overlapping puzzle pieces. On the left are the contours of an image. On the right are images showing the contours on the left superimposed with images of the final hypotheses filled with patterns so that the various final hypotheses may be distinguished from each other. Additional results of running the algorithm on images of puzzle pieces may be found in Appendix B.

than the algorithm, however, they also averaged more false alarms per image. Humans do better with less boundary visible than the algorithm, presumably because they use other information in addition to boundaries.

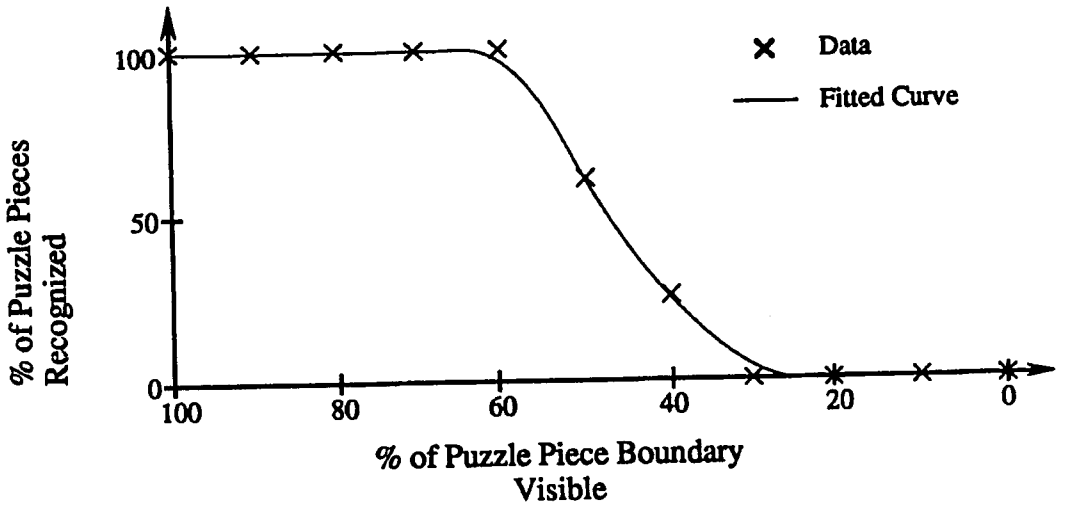


Figure 4.42. Fraction of Puzzle Pieces Recognized Correctly vs. Fraction of Boundary Visible. The performance statistics were gathered over a sample of six images.

4.5.6.5 Switch Parts

Fig. 4.43 shows a representative example of running our algorithm on an image of overlapping switch parts. Additional results may be found in Appendix B. Fig. 4.44 gives a plot of the percentage of the switch parts that were recognized correctly versus the percentage of part boundary visible. As can be seen from both Figs. 4.43 and 4.44, the algorithm had more difficulty recognizing the switch parts than it did recognizing the puzzle pieces. Table 4.3 summarizes the false alarm statistics on the images of the switch parts.

The decision region for the switch parts is given by (4.8) and the following list of parameter values: $T_1 = .3$, $T_2 = .3$, $T_3 = .4$, $\beta = .6$. The runtime of the algorithm on images containing the switch parts averaged 1.5 seconds, again from the step of feature detection onward.

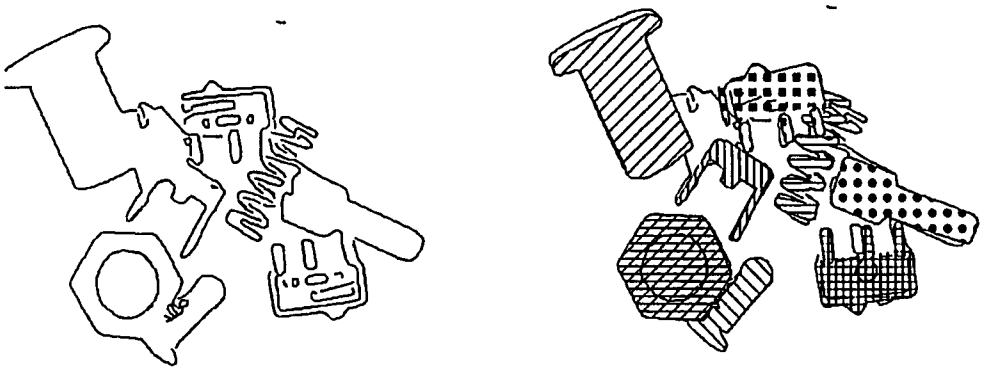


Figure 4.43. Above is a sample of the results of running the algorithm on the images of the switch parts. This figure is analogous to Fig. 4.41. Additional results examples of running the algorithm on images of the switch parts may be found in Appendix B.

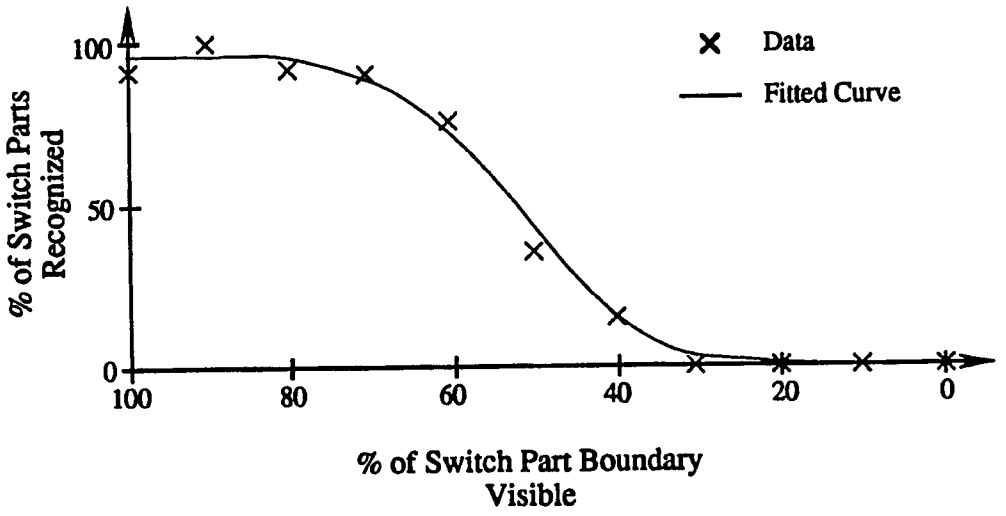


Figure 4.44. Fraction of Switch Parts Recognized Correctly vs. Fraction of Boundary Visible. Performance statistics for this figure were gathered from a sample of fourteen images.

4.5.7 Conclusion

We have presented a new procedure for 2-d partially visible object recognition. The neighborhoods of critical points were employed as the fundamental features. The heart

of the method was the use of a k-d tree for fast feature matching. The use of the k-d tree was made feasible by applying the Karhunen-Loève expansion to the feature vectors to reduce the data in them by an order of magnitude. Experiments were conducted on two sets of real objects, jigsaw puzzle pieces and switch parts, to get an idea of the accuracy, robustness, and efficiency of our algorithm. A high degree of accuracy was obtained for objects having more than 50-60% of their boundary exposed. Objects that were more heavily occluded were still recognized much of the time, but success was less certain. Although a direct comparison is inappropriate, it is interesting to compare this with the success one would expect from the analogous problem of recognizing a sentence with half of the letters and spaces missing.

The results of the experiments we have conducted and our experience developing the algorithm has led to an interesting conclusion. Observe first that in a few of the images in Fig. 4.43, the spring was found in the image shifted one or more cycles from the correct position. This is not surprising since all of the critical points along the side of the spring are very similar and generate many spurious hypotheses which place the spring shifted from where it should be. Our algorithm occasionally chooses one of the spurious hypotheses because it may just happen to adjoin a section of boundary from another part, thus making Q_b large enough to pass the false hypothesis over the correct one. In fact, this scenario also leads to most of the false alarms in the experiments. Interestingly, all of these false alarms as well as most of the misplacements of the spring could easily be eliminated if some simple segmentation information was employed in addition to just the shape of the edge contours. In particular, if the background region was known, then these problems could often be eliminated since, in many cases, such false (or poor) hypotheses will have large sections of their contour deep in background with no other contours nearby. This information could be used to weaken those hypotheses, making the correct one more likely to be chosen as a final hypothesis. We believe that employing segmentation information will be necessary to solve the problem in 3-d of partially visible

object recognition. Our algorithm currently uses no information about the is background.

Finally, we note that our recognition system could be extended in a straight forward manner to the recognition of scaled objects if a scale-invariant feature vector that is not sensitive to noise could be found. There are many possibilities employing multiple points. We have found a method that normalizes scale using local curvature information at a single critical point to yield a scale invariant feature vector encoding the local shape of the object [GM88]. These features show promise for use in recognition of both scaled 2-d and 3-d objects. Using these observations, we are currently working to extend the method presented here to the domain of 3-d partially visible objects. In addition to the edge-based features described here, it is likely that other attributes of objects will be useful in recognition. The k-d tree matching technique that we have described is sufficiently general to accommodate such extensions with ease.

4.6 Chapter Summary

In this chapter we have fleshed out part of the Cyclops framework for 3-d object recognition. We have also investigated the least understood aspects of the framework to prove that the overall framework is feasible. In particular, we implemented a 2-d recognition sytem that uses the approach to hypothesis generation of the Cyclops framework. This work empirically showed the effectiveness and efficency of this approach. We showed that, in the absence of the object-attached feature assumption, multiview models become a key component of the hypothesis generation process, and describe a hybrid multiview model that combines a multiview feature representation and a 3-d representation. We also analysed the interplay between the selection of features and hypothesis generation. We described the computation features formed from pairs of critical points and inflection points that possess many advantageous properties for hypothesis generation. Finally, we implemented a rule-based approach to contour grouping that is based on continuity of

contours and discussed how to assign a perceptual significance score to features using this information.

CHAPTER 5

MODEL-BASED VIEWING PARAMETER ESTIMATION AND TRACKING

Precise, viewing parameter estimation is a critical part of any model-based recognition or tracking system. This chapter discusses an approach to this problem that addresses the limitations of previous methods. Our approach, called *Attidude Estimation by Feature Modulated Attractors*, or AEFMA, is based on optimizing a carefully constructed measure of the disparity in shape of edge contours predicted from a 3-d model and the shape of edge contours detected in the image. Experiments, described in Section 5.9, show that AEFMA has a wide range of convergence. In spite of grossly erroneous initial estimates of the viewing parameters, AEFMA is able to determine the correct viewing parameters. Further, this performance is achieved even in the presence of clutter, distortion, and occlusion. Finally, AEFMA achieves greater generality with respect to the shapes of the objects and models with which it is able to operate than do previous methods. In particular, AEFMA makes no use of the object-attached feature assumption. Thus, AEFMA is freed from the limitations that result when the object-attached feature assumption is invoked.

The above discussion implies that AEFMA would be a robust addition to any 3-d object recognition or tracking framework. In particular, AEFMA is an integral part of the overall Cyclops 3-d recognition framework. Fig. 5.1 shows where AEFMA fits into Cyclops. Precise attitude estimation is important in Cyclops since, as explained in Chapter 4, the hypothesis generator can only guarantee the accuracy of the viewpoint

parameters to within the spacing of the neighboring representative views on the viewing sphere. Ideally, in order to reduce the storage requirements of the multiview models, we desire that the spacing of the representative views over the viewing sphere be as large as possible while maintaining the performance that we require. Thus, if we are able to accurately refine the hypothesis generator's estimate the viewing parameters even when the initial viewing parameters may be poor, the spacing of the representative views over the viewing sphere may be made very sparse. In fact, if the viewing parameter estimation procedure is good enough, a "brute force" recognition paradigm, normally dismissed as too computationally expensive, becomes feasible. This paradigm, a variation of the "try all possible models at all possible poses" approach, becomes feasible because we really do not need to try all possible models at all possible poses; we need to try just enough of them to insure that if the object is present in the image, the viewing parameter estimator will be able to determine its viewing parameters. Then, initial guesses whose viewing parameters could be successfully refined are considered to be recognized in the image. While AEFMA does make this approach feasible, providing a good initial hypothesis, as in Cyclops, leads to a much more efficient approach. Nevertheless, it is remarkable that AEFMA renders such a brute force approach feasible.

Clearly AEFMA is an important component of the overall Cyclops framework. By itself, however, AEFMA is nearly a stand-alone, 3-d, model-based tracking system. The model-based tracking paradigm is illustrated in Fig. 5.2. There, we assume that at time t we possess a current "best estimate" of the pose and motion of a model. Of course, this estimate will not agree exactly with the new frame of image data that we also receive at time t. Our problem of updating the best estimate can be broken into two subproblems, shown in Fig. 5.3: using the new frame of data, and the current best estimate of the model's pose and motion parameters to determine a new estimate of the pose parameters at t using the former best estimate of the pose along with the new frame of image data; and using the corrected pose parameters to reestimate the current motion parameters, thus

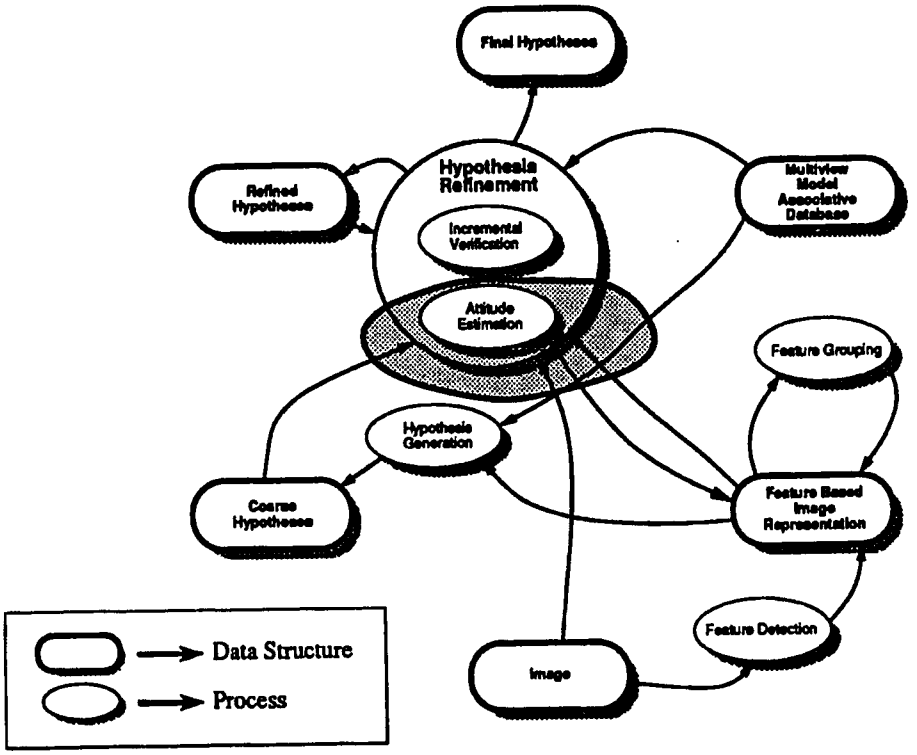


Figure 5.1. Where AEFMA fits into the overall Cyclops framework.

enabling us to determine a new best estimate of the model's pose and motion parameters at time $t + dt$. The first problem, that of reestimating the current pose parameters at time t, is precisely the problem that AEFMA solves. The second problem, that of updating the best estimates of the motion parameters and producing a new best estimate of the model's pose and motion at $t + dt$, is beyond the scope of this thesis. However, some work has been done along these lines. For example, Dickmanns [Dic87, Dic88] describes a promising approach based on Kalman filtering.

5.1 Overview of Attitude Estimation by Feature Modulated Attractors (AEFMA)

At the heart of AEFMA is a carefully constructed *Composite Disparity Function* (CDF) that measures the disparity between the shape of image edge contours and the shape of edge contours predicted from the model. Then, given a set of initial viewing parameters, the viewing parameters of the model are adjusted to minimize the disparity

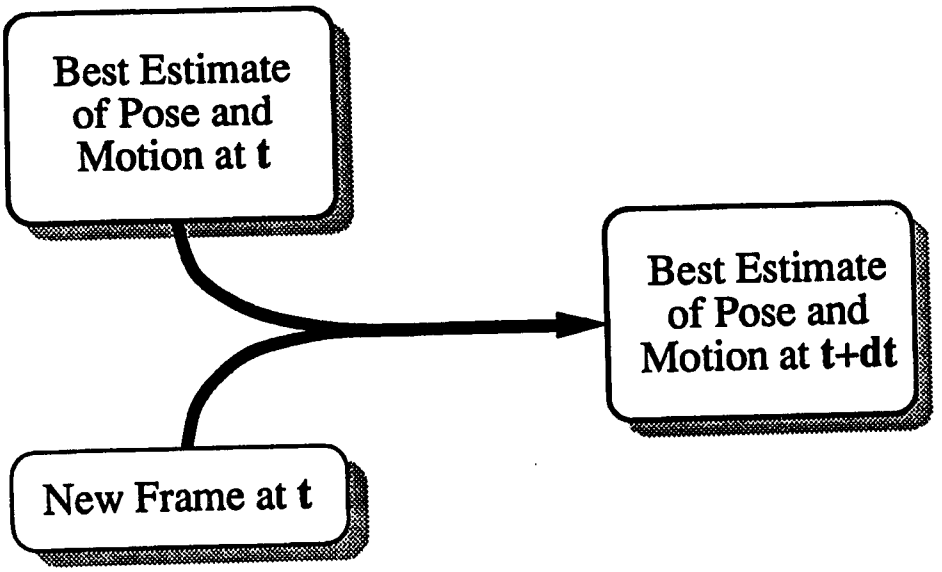


Figure 5.2. The task of model-based tracking is to provide a best estimate of the pose and motion of an object at any time t based on a sequence of frames of sensory data. In particular, given the last best estimate of an objects state, i.e., its pose and motion at time t, and a newly acquired frame of data, update the estimate of pose and motion to agree with the new data subject to the physical constraints on the moving object. Then, extrapolate into the future to time $t + dt$ to determine the best estimate of the object's state in the following frame.

function. From this description, we see how AEFMA avoids using the object-attached feature assumption. The composite disparity function (CDF), whose computation will be described shortly, is a function *only* of the 2-d image edges and the 2-d predicted model edges. AEFMA never makes any direct correspondence or comparison with the surfaces of the 3-d model: rather it only uses the 3-d model as a means to predict what the edge contours of the object will look like when viewed under a particular set of viewing parameters.

Viewed abstractly, AEFMA can be thought of as a *contour matcher* that adjusts the parameters of a model of the edge contours until the appearance of the contours predicted by the model is most similar to the edge contours detected in the image. Thus, AEFMA is *decoupled* from the particular nature of the models that it uses. Contrast this to the

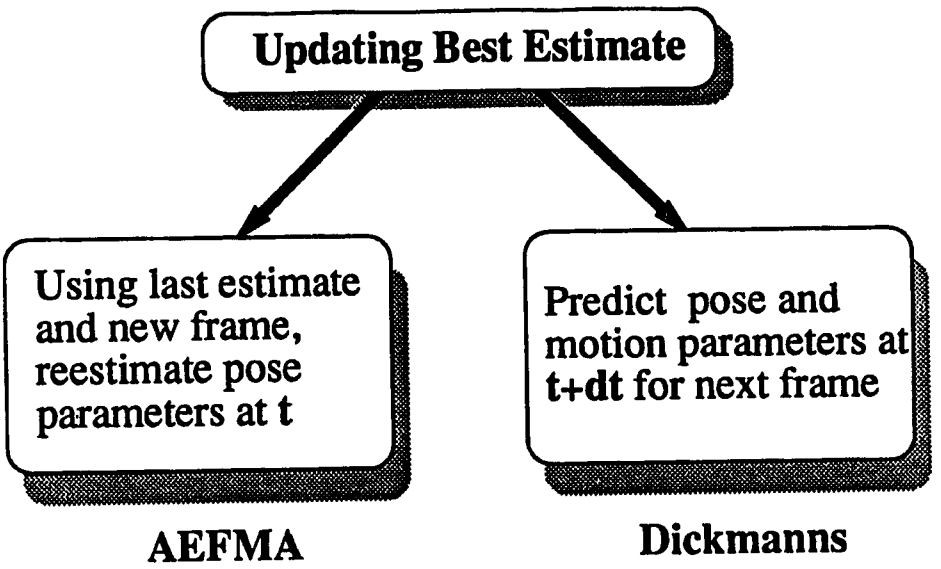


Figure 5.3. Model-based tracking can be split into two independent subproblems. The first is to use the former best estimate of the current pose at t as a starting point to find the actual pose parameters at t. This problem is addressed by AEFMA. The second problem is to use the new estimate, and the previous estimates, to extrapolate the position and motion of the object to $t + dt$ for use as the initial pose estimate when the next frame arrives. This problem is being explored by, among others, Dickmanns [Dic87, Dic88]

tight coupling between the 3-d models and 2-d features that exists in algorithms that employ object-attached features (see Chapter 3). This decoupling allows AEFMA to be easily generalized to other types of models, such as models of articulated objects, and deformable objects.

Fig. 5.4 depicts AEFMA's approach. At the bottom of the figure is the image plane, where the image edge contours of a space shuttle obscured by a satellite have been detected. At the top of the figure is the 3-d model. A viewing transformation is applied to the 3-d model to obtain a prediction of the edge contours that appear from a particular viewpoint, as shown in the "prediction" plane in the center of the figure. In AEFMA, both the predicted edge contours and the detected edge contours are represented by a shape representation consisting of a set of *shape primitives*. Each shape primitive consists of a description of the geometric relationships between two points and the shape of the

contours near each point. In Fig. 5.4, the shape primitives are shown by the dotted lines between two points on an edge contour. For clarity, only one predicted model primitive and two detected image primitives are shown in the figure. As will be described, there are typically many more. The essence of AEFMA is that the viewing parameters are adjusted to minimize a measure of the overall difference, or disparity, in the shape of the predicted edge contours and the detected edge contours. This measure of the overally disparity in shape between what is predicted and what is actually observed is the CDF. The CDF can be viewed as an energy potential function resulting from image pseudo-forces acting on the model. The total force acting on the model is the result of attractive pseudoforces acting between the predicted shape primitives, as in the middle plane of Fig. 5.4, and the image shape primitives, as in the bottom plane of Fig. 5.4. The attractive pseudoforces between the shape primitives are shown by the cones of ellipses in Fig. 5.4. The large range of convergence of AEFMA is achieved by *modulating* the forces by the similarity in their attributes, i.e., if the primitives possess similar attributes, the attractive forces are powerful, whereas if they possess dissimilar attributes, the attractive force acting between them is weak. This is the origin of the phrase "feature modulated attractors" in AEFMA's name. Minimizing the CDF is equivalent to finding the equilibrium among all the forces. Since the most similar portions of edge contours are those that exert the most powerful forces, "equilibrium" results in a good fit for the image contours that can be best explained by the model at some pose.

To be more specific about the description in the previous paragraph, let $\mathcal{S}_m$ be the shape representation of the predicted edge contours and $\mathcal{S}_i$ be the image edge contours. Then, the composite disparity function is a real function $\mathrm{CDF}(\mathcal{S}_i, \mathcal{S}_m)$. We compute $\mathrm{CDF}(\mathcal{S}_i, \mathcal{S}_m)$ by summing the result of scoring the disparity between each pair of an image shape primitive with a predicted shape primitive. The disparity between individual shape primitives p^i describing an image edge contour, and p^m describing a predicted edge contour, is measured by the *interfeature disparity function* (IDF). The IDF is a negative

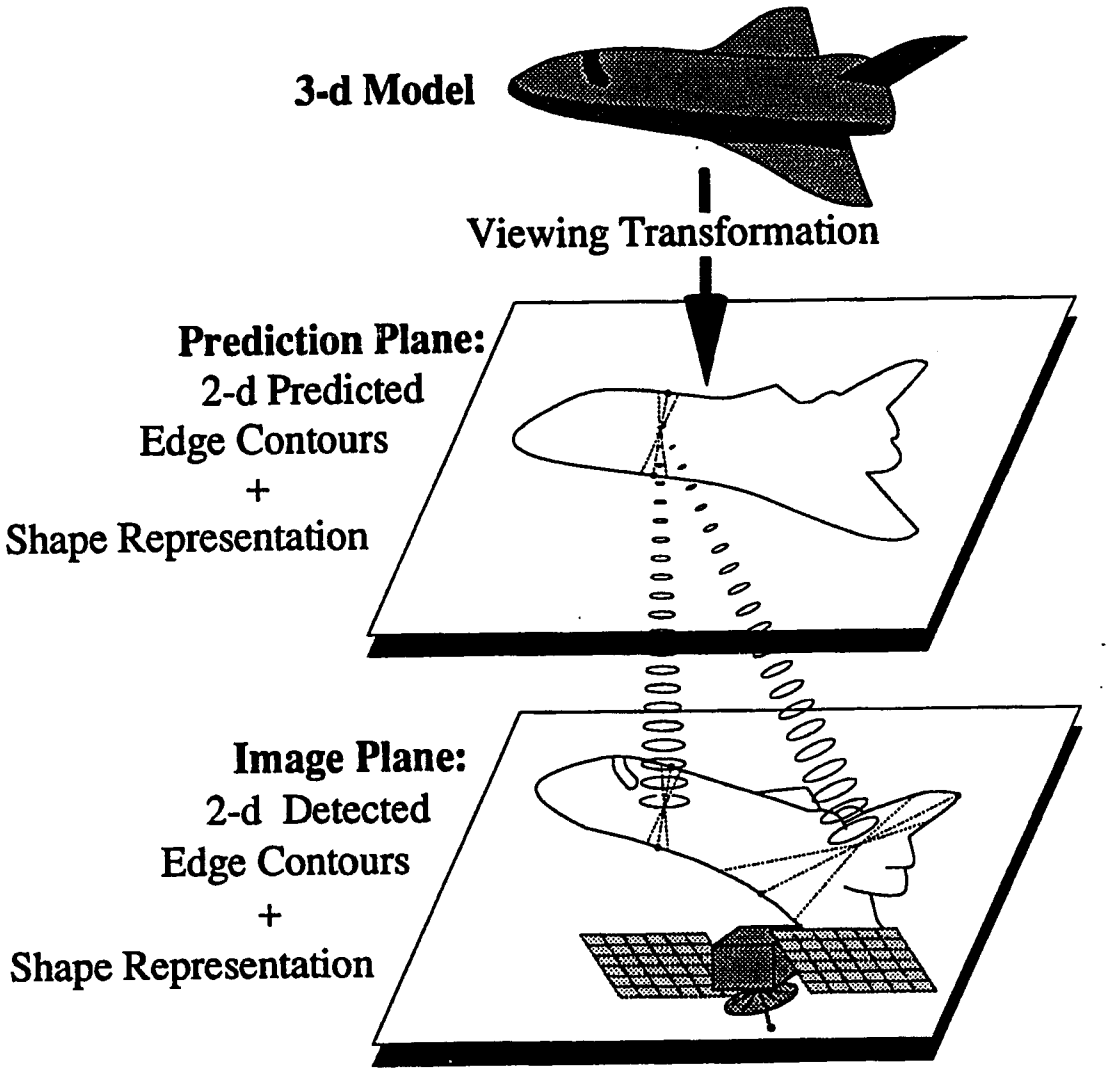


Figure 5.4. For a particular viewing transformation, the 3-d model is used to predict which edge contours will be present when the object is viewed (prediction plane). Similarly, the edge contours in the image are detected. The shape of the edge contours, both predicted and detected, are represented by a set of shape primitives, described in the text. For each model primitive in the prediction plane, such as the one shown, a disparity measure, called an IDF, is calculated for each image primitive in the image plane, such as the two shown. All such pairs are summed to form a composite disparity function which can be used to optimize the viewing parameters to obtain the viewing parameters resulting in the best match between the shape of the predicted edge contours and the shape of the image edge contours.

function that approaches zero when the primitives are not similar in shape, and approaches negative one as the primitives become identical. If we assume for the moment that the relative 2-d position, orientation, and size of a pair of shape primitives does not change, the magnitude of the IDF is increased if the *shapes* of the detected primitive and the predicted primitive become more similar, and is decreased if the *shapes* of the primitives become less similar. Thus, as described above, the magnitude of the IDF is *modulated* by the disparity in shape between the primitives. We capture the CDF mathematically by the following equation:

$$\mathrm{CDF}(\mathcal{S}^p, \mathcal{S}^i) = \sum_{j=1}^{N} \sum_{j=1}^{M} \mathrm{IDF}(p_j^p, p_k^i, \sigma) \tag{5.1}$$

where N is the number of primitives in the predicted shape representation, M is the number of primitives in the image shape representation, p_j^p is the j^{th} predicted shape primitive, and and p_k^i is the k^{th} image shape primitive.

A key aspect of AEFMA that we have not yet mentioned is its coarse-to-fine nature. The CDF has a set of parameters, embodied in the σ vector, that allows the specificity of the CDF to be adjusted. Precisely how this is accomplished will be discussed shortly. For now, we simply note that setting the components of the σ-vector to large values results in a CDF that permits pairs of primitives to make significant contributions to the CDF even though the attributes of the predicted primitive and detected primitive differ considerably. This prevents a few pairs of erroneously matched primitives whose attributes are similar by chance to overcome the effects of many pairs of correctly matching primitives whose attributes are less similar. On the other hand setting the component of the σ-vector to small values results in the CDF being very specific, i.e., only image shape primitives whose attributes match a model shape primitive very closely are allowed to contribute significantly to the CDF. In the early stages of the algorithm, AEFMA sets the components of the σ-vector to large values since the initial estimates of the viewing parameters may be poor, implying that the attributes of a correctly matching pair of shape primitives may differ considerably. If the componenents of the σ-vector were set to small values

initially, the correctly matching shape primitives in the image would be wrongly excluded from making a significant contribution to the CDF. In such a case, since the correctly matching shape primitives would not be contributing to the CDF, it is very unlikely that the optimization procedure could converge to the correct values of the viewing parameters. Thus, the components of the σ-vector are set to large values during the early stages of the algorithm. However, if the magnitude of the σ-vector is left large, the localization of the minima suffers, due most strongly to the influence of incorrectly matching shape primitives, and less strongly to the fact that a broader minima is more difficult to localize accurately. This results in poor estimates of the viewing parameters. A more effective strategy is to progressively reduce the values of the σ-vector's components. This reduces the influence of incorrectly matching primitives on the CDF, and enhances the influence of correctly matching primitives on the CDF, thus improving the location of the minima of the CDF, and, consequently, the resulting estimates of the viewing parameters.

5.1.1 Pseudo-Code Description of AEFMA

Having discussed the function of AEFMA from a high level, we summarize the discussion thus far with a description in terms of C-like pseudo-code. As described earlier by Eq. 5.1, the CDF is a function of the shape of the edge contours detected in the image and the shape of the edge contours predicted from the model. Implementationally, as shown in the pseudo-code in Fig. 5.5, the CDF routine is treated as a function of the viewing parameters. The image shape representation is accessed by the CDF function as a variable pointing to a *static*, i.e., *not dynamic* data structure. Similarly, the CDF can access the model, and uses the model and the viewing parameters to predict the edge contours and compute their predicted shape representation. For those who are not familiar with C, a static variable is a variable that is accessible to a routine, and may be accessible to certain other routines as well. In Fig. 5.5 we assume that the static variables have been set before the CDF routine is called. Further, comments are italicized.

```
Declarations (previously set variables)

static mode_type                    model;
static primitive_database_type      detected_shape_database;
static sigma_vector_t               sigma_vector;

CDF(viewing_parameters)
      viewing_parameter_type        viewing_parameters;
{
      primitive_database_type       predicted_shape_database;
      float                         partial_disparity;
      float                         composite_disparity;
      shape_primitive_type          dsp, psp;

      Initialize the composite disparity.

      composite_disparity = 0.0;

      Use the viewing parameters and the model to predict the appearance of the edge contours
      from the the view specified by the viewing parameters.

      predicted_shape_database = build_shape_primitive_database(model,viewing_parameters);

      Form all possible pairings of a detected shape primitive with a predicted shape primitive
      and sum the contribution from each pairs interfeature disparity function (IDF).

      for each shape primitive, psp, in predicted_shape_database do
      {
            partial_disparity = 0.0;
            for each shape primitive, dsp, in detected_shape_database do
            {
                  partial_disparity += IDF(psp, dsp, sigma_vector);
            }
            composite_disparity += partial_disparity;
      }
      return(composite_disparity);
}
```

Figure 5.5. Conceptual seudocode description of the computation of the CDF.

The pseudocode description of the computation of the CDF in Fig. 5.5 is conceptually accurate. There is, however, one aspect of it that does not entirely reflect the way AEFMA is implemented. In particular, the pseudocode indicates that the IDF is computed for all the possible pairs of an image shape primitive with a predicted shape primitive. In fact, as we will show, most of these pairs result in negligible IDF's. Thus, the execution of AEFMA could be vastly speeded up if the IDF was computed *only* for those pairs of

```
Form all possible pairings of a detected shape primitive with a predicted shape primitive
and sum the contribution from each pairs interfeature disparity function (IDF).

for each shape primitive, psp, in predicted_shape_database do
{
      partial_disparity = 0.0;
      significant_contributors = get_significant_contributors(detected_shape_database, psp);

      for each shape primitive, dsp, in significant_contributors do
      {
            partial_disparity += IDF(psp, dsp, sigma_vector);
      }
      composite_disparity += partial_disparity;
}
```

Figure 5.6. The pseudocode for implementing the CDF efficiently.

primitives that make significant contributions to the CDF, i.e., have non-negligible IDF's. We, in fact, have done this. To reflect this, the pseudocode for the nested loop in Fig. 5.5 should be written as shown in Fig. 5.6. The key difference is the addition of a call to the function **get_significant_contributors** which accesses the detected_shape_database (declared in Fig. 5.5). This call returns only the significant contributors to the CDF. The inner loop then iterates only over these primitives to compute the partial_disparity. Since the number of significant contributors is far smaller than all possible detected primitives, this greatly speeds the computation of the CDF, assuming, as we will show, that **get_significant_contributors** can itself be implemented efficiently. As will be shown later, the database access function **get_significant_contributors** can be implemented very efficiently. Further, we will show that the number of significant contributors is very small, resulting in a drastic speedup of the computation of the CDF.

Having described the computation of the CDF, Fig. 5.7 shows the pseudo-code for the top level of the AEFMA algorithm. It is at this level that the optimization and the coarse-to-specific nature of the algorithm can be seen. As can be seen from Fig. 5.7, the values of the components of the σ-vector is reduced by a constant factor after each iteration. Thus, referring to Fig. 5.8 (a), the minimum of the CDF is initially poorly localized, but the "hole", i.e., the range of viewing parameters where the correct minimum dominates

any local minima, will be large, increasing the likelihood that the optimization procedure will be able to locate the minimum starting from the initial estimate of the viewing parameters. After an iteration occurs, and the components of the σ-vector are reduced in value, the minimum of the CDF becomes more accurately localized. At the same time, the size of the "hole" has become smaller, as shown in Fig. 5.8 (b). However, since the viewing parameters have now been improved, assuming that the reduction factor is not too large (1.5-2.5 is typical), then the initial values of the parameters will fall within the confines of the smaller "hole", as shown in Fig. 5.8 (c), permitting the optimization procedure to successfully converge to the improved minimum of the CDF, an so on until the length of the σ-vector becomes too small.

5.1.2 Continuous Versus Discrete Optimization

At this point we examine the issue of whether AEFMA should be based on a continuous or a discrete optimization method. We have chosen continuous optimization because most continuous optimization methods take advantage of the continuity and smoothness of the functions they attempt to minimize, allowing them to search a far smaller region of the function's domain. This improved efficiency is not without cost, however. Specifially, since continuous optimization methods use only local information, such as the function's values, gradients, and Hessians, at various points, they find usually find the local minimum of the function that is nearest to the supplied starting point. On the other hand, discrete methods, such as simulated annealing [Rut89, PFTV86], dynamic programming [Bel57, Dre77, AWJed], branch and bound [LW66], and heuristic searching [Pea84, Nil80], tend to be more global, usually being able to find the best minima over a wide range of the function's domain. Unfortunately, since discrete methods do not use the powerful information inherent in the knowledge of the function's n^{th} order continuity, they typically must do much more searching, and therefore tend to be more inefficient than continuous methods. This is especially true if the cost of evaluating the function is

```
Global declarations

image_type                  image;                      Input
model_type                  model;                      Input
sigma_vector_type           sigma_vector;               Input
viewing_parameter_type      initial_viewing_parameters; Input

primitive_database_type     detected_shape_database;    Image shape.
viewing_parameter_type      viewing_parameters;         Current viewing parameters.

End of declarations, beginning of executable code.

Find edge contours and build shape primitive database.

detected_shape_database = build_primitive_database(image);

Set the static parameters so that the CDF routine may access them.

set_static_variables(model, detected_shape_database, sigma_vector);

Set the initial view.

viewing_parameters = initial_viewing_parameters;

do
{
    Find the viewing parameters that minimize the composite disparity function (CDF)
    by starting at the view specified by viewing_parameters.

    viewing_parameters = minimize_over_view_parameters(viewing_parameters, CDF);

    Reduce each of the components sigma_vector by a factor.

    sigma_vector = reduce_sigma_vector_components(sigma_vector);
}
while (sigma_vector) ≠ final_sigma_vector);
```

Figure 5.7. Top level pseudocode of AEFMA.

high. Another consideration is that the viewing parameters form a continuous space. In order to apply a discrete optimization method, the parameter space must be discretized. The best way to accomplish this is far from obvious.

Some researchers have attempted to reap the benefits of the efficiency and fast convergence of continuous methods and the immunity to shallow local minima of discrete methods. They have attempted to do this by forming hybrids of continuous and discrete methods. For example Solina *et al* [Sol87, SB90] have modified the Levenberg-Marquardt method [PFTV86, Sca85] by adding Poisson noise to the value of the function in the testing phase of the Levenberg-Marquardt method where it is deciding whether to accept

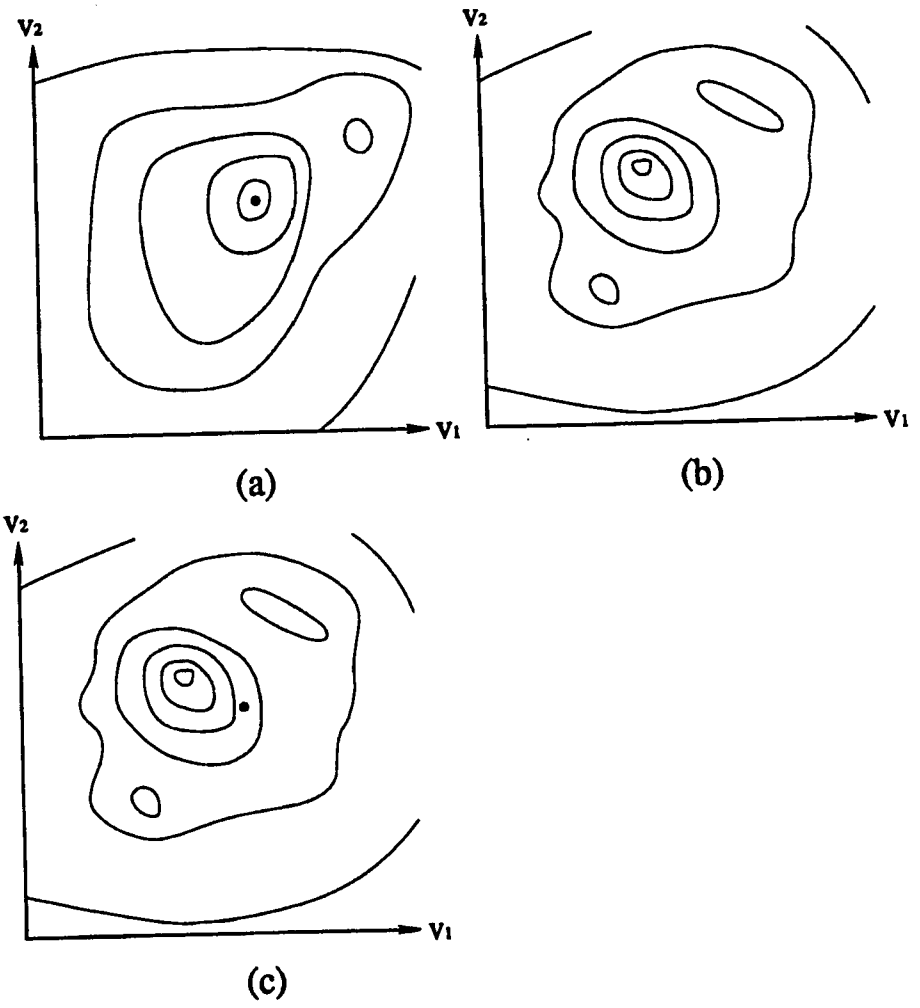


Figure 5.8. The sequence of figures (a), (b), and (c) illustrate why AEFMA's coarse-to-fine strategy is effective. In (a) is a contour plot of a hypothetical CDF resulting from a σ-vector with large values, while (b) shows the CDF after the values of the σ-vector have been halved. The minimum of the CDF in (a) is marked with a dot. In (c) is the contour plot of (b) with the minima of (a) (the dot) superposed. Notice that the minimum has moved, but the minimum of (a) remains within the valley that would allow a continuous optimization procedure to find the minimum of (b) using the minimum of (a) as the starting point.

the newly calculated parameter vector. Thus, the algorithm is permitted to take steps that result in an increased value of the objective function with probability that depends on the

size of the λ parameter of the Poisson distribution. With this modification, the author claims that the algorithm is able to avoid shallow local minima.

The problem with the hybrid approaches, such as the example above, is that they have no theory to guide their application. In particular, the method above cannot be guaranteed to converge, nor can its behavior be easily predicted. As a case in point, take the hybrid Levenberg-Marquardt method described above. There, if the value of the Poisson λ parameters is not chosen carefully, the algorithm may kick out of a correct minima into a neighboring, less good, local minima, and never be able to find its way back, while a subsequent run of the algorithm may be successful.

We have adopted a contrasting strategy in AEFMA, indeed, in Cyclops as a whole. First, as to whether to use discrete or continuous optimization, we deem the advantages of continuous optimization to outweigh the disadvantages, if ways can be found to reduce the likelihood of being trapped in a local minimum to acceptable levels. Further, in contrast to the hybrid described above, we have insisted on using well-understood numerical optimization techniques. Rather than attempt to change the optimization procedure, we have taken the approach of insuring that the composite disparity function is easily optimized (i.e. by insuring that there are not any strong local minima near to the correct minima of the CDF). Further, Cyclops' hypothesis generator insures that the starting point of the optimization is near enough to the correct solution that the procedure will be very likely to find the correct minima. This allows us to use an optimization procedure that is well understood theoretically, and, most importantly, can be guaranteed to converge consistently from run to run.

5.2 Design of the Interfeature Disparity Function

As shown by Eq. 5.1 and Figs. 5.5, 5.6, and 5.7, the interfeature disparity function, or IDF, plays a key role in AEFMA. We now consider the design of the IDF.

The IDF is a real function mapping $\mathcal{P}^2 \rightarrow [-1, 0]$, where $\mathcal{P}$ is the set of shape primitives. A shape primitive can be viewed exactly like a feature, i.e., as a vector of attributes. Indeed, the following section describes how AEFMA's shape primitives are very similar to the point-pair features used by Cyclops' hypothesis generation process. Like those features, all of the attributes describing AEFMA's shape primitives are real numbers. Thus, for simplicity, we will treat the IDF as a function mapping $\Re^{2N} \rightarrow [-1, 0]$, where N is the dimension of the attribute vector. For the remainder of the discussion, it will be useful to think of each IDF as a function of the predicted primitive's shape attributes alone, while the image primitive's attributes are "measured" constants.

We have identified ideal candidates for the IDF: multidimensional Gaussians of the form

$$\mathrm{IDF}(\mathbf{p}^i, \mathbf{p}^p) = \exp\left[\sum_{k=1}^{N}\left(\frac{p_k^i - p_k^p}{\sigma_k}\right)^2\right], \tag{5.2}$$

where $\mathbf{p}^i$ is an image shape primitive attribute vector whose components are p_k^i, $\mathbf{p}^p$ is a predicted shape primitive attribute vector whose components are p_k^p, and $\sigma_k > 0$ are the components of the σ-vector. Recall that the components of the σ-vector control the width of the multidimensional Gaussian. What considerations lead to the choice of Gaussians for the IDF? Recalling that the CDF is the summation of many IDF's, the factors in Table 5.1 influence the selection of multidimensional Gaussians for the IDF.

Factor (a) in Table 5.1, that the IDF should be minimal when the attributes of the primitives are identical is the basis our approach: when the predicted shape matches the observed shape, the function values are minimal, and the viewing parameters are correct.

Factor (b) in Table 5.1, that the IDF should avoid introducing any spurious local minima into the CDF, is very important. If local minima are introduced in the CDF,

Factor	Description
(a)	The IDF should be minimal when the attributes of the primitives are identical.
(b)	The IDF should introduce few local minima into the CDF.
(c)	The IDF should be approximately parabolic near the minimal value.
(d)	The IDF should approach zero for primitives whose attributes are very dissimilar.
(e)	The IDF should smoothly blend the two extremes.
(f)	The IDF should be parameterized so that the size of the parabolic region is easily adjusted

Table 5.1. Factors influencing the design of the IDF.

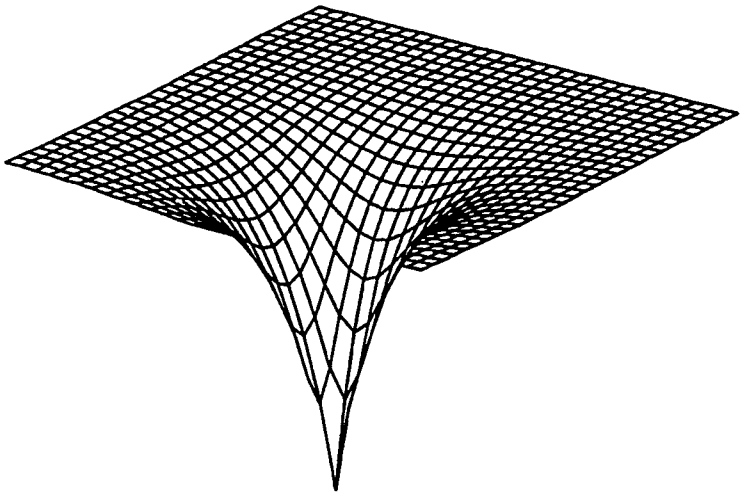


Figure 5.9. A candidate for an IDF with a curvature singularity. This function, $-\exp(-ar)$, where r is the distance from the center of the function, is analytic everywhere except at it center where a singularity with infinite curvature exists. In the text, we show that such functions are not suitable candidates for use as the IDF.

especially in the neighborhood of a correct solution, then there is no way that an efficient continuous optimization procedure can succeed in finding the global minimum reliably. This leads to two conclusions. The first is that the IDF cannot possess any non-global local minima of its own. Were it to possess non-global local minima, it is possible that the local minima, produced by a strongly, albeit incorrectly matching pair of primitives, would overpower the primary minima of a large number of less strongly, but correctly matching,

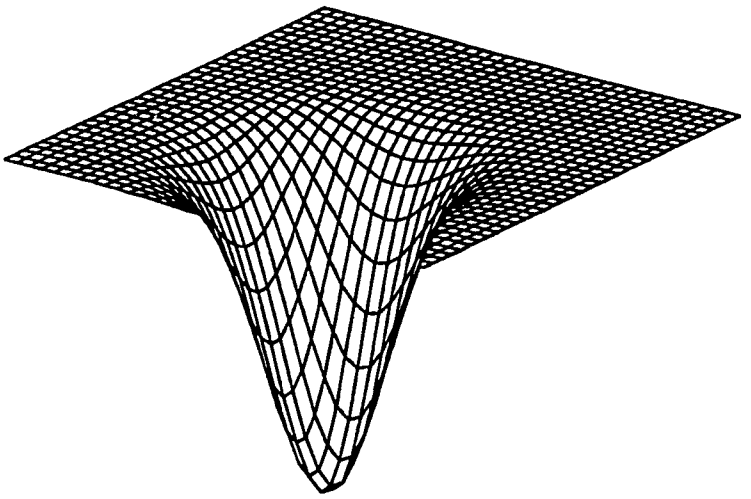


Figure 5.10. A 2-d Gaussian of the form $-\exp(-br^2)$, where r is the distance from the center of the function. This function is analytic everywhere. We have chosen the values of a in the function $-\exp(-ar)$ in Fig. 5.9 and b in the function $-\exp(-br^2)$ shown in this figure so that the two functions have equal second moments, a measure of their widths. This allows the functions to be compared in later figures. In the text we show that higher-dimensional versions of this function are ideal for use as the IDF.

pairs of primitives. The second conclusion is that the IDF may have no singularities in curvature. This is somewhat more subtle than the first, and is best illustrated graphically. Fig. 5.9 shows a 2-d candidate for an IDF of the form $-\exp(-ar)$, where r is the radial distance from the center, and $a > 0$ is a factor that controls the width of the function. This function has a curvature singularity at its center, as seen by the pointy nature of the function there. Fig. 5.10 shows another 2-d candidate for an IDF. This function is a multidimensional Gaussian of the form $-\exp(-br^2)$, where, again, r is the radial distance from the center, and $b > 0$ is a scale factor controlling the width of the function. Multidimensional Gaussians are analytic, i.e., all orders of derivatives exist. Thus, it has no curvature singularities. The parameters a and b have been adjusted so that the second moments of the two functions (a measure of their widths) are the same. This permits the

functions to be compared directly. Fig. 5.12 shows what happens as three symmetrically placed functions of the type shown in Fig. 5.9 are summed together at varying spacings. No matter how closely the functions approach each other, as they would be in the case of a number of strongly matching pairs of primitives, local minima remain. In contrast, Fig. 5.11 shows the result of summing three symmetrically placed 2-d Gaussians. In this case, local minima appear only when the functions are very far apart, as in Fig. 5.11 (a). When the minima are moved closer to each other, yet still quite far apart (about one sigma), as in Fig. 5.11 (b), the Gaussians' individual minima blend the local minima into a single global minimum. This property, which also hold for higher dimensions and large numbers of component functions, is precisely what we seek in the IDF since we desire to suppress the minima of the individual IDF's and produce a global minimum that aggregates minima of the individual IDF's. Further, as the width of the Gaussians are increased, the maximum distance where the minima of the IDF's blend into a single minimum increases proportionately. This is important, as we desire a greater suppression of local minima when the components of the σ-vector take on large values.

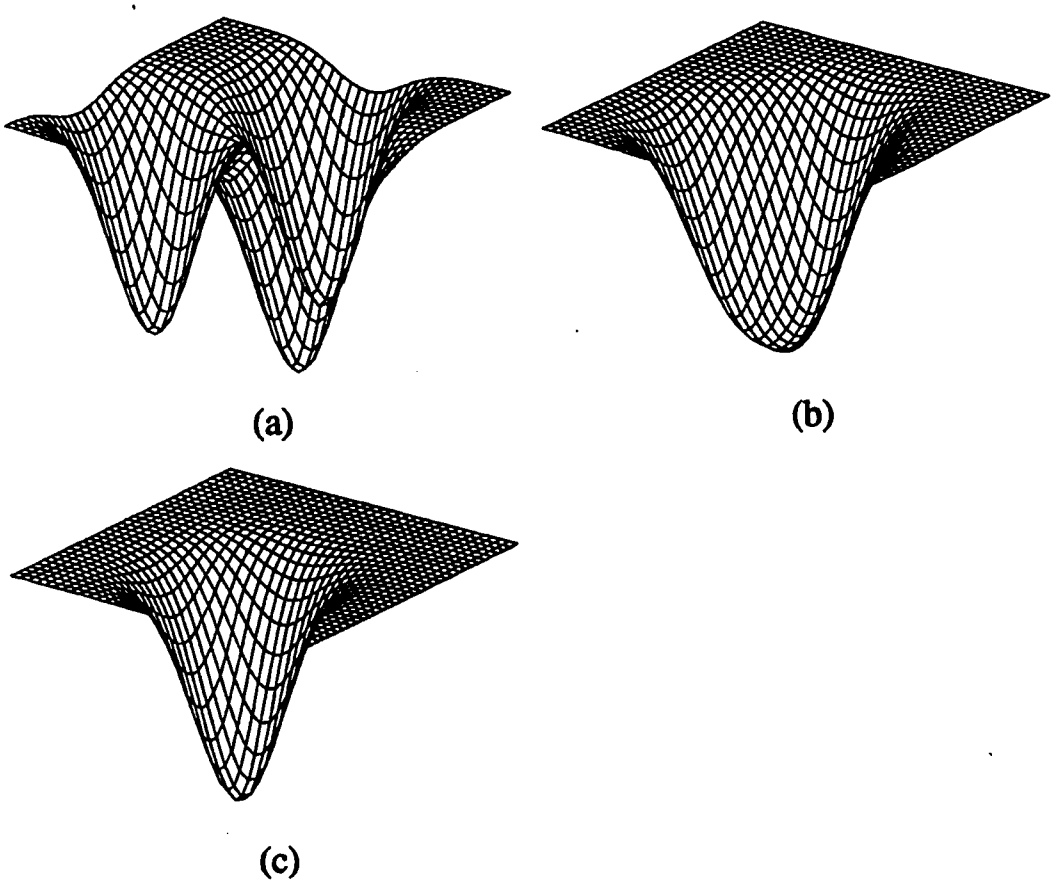


Figure 5.11. The sum of three symmetrically placed 2-d Gaussian IDF's. In (a) the functions are far enough apart that there are three local mimima. In (b) however, even though the functions are still quite far apart, their minima have merged into a single minima. In (c), the functions are still closer, and, of course, a single minima remains.

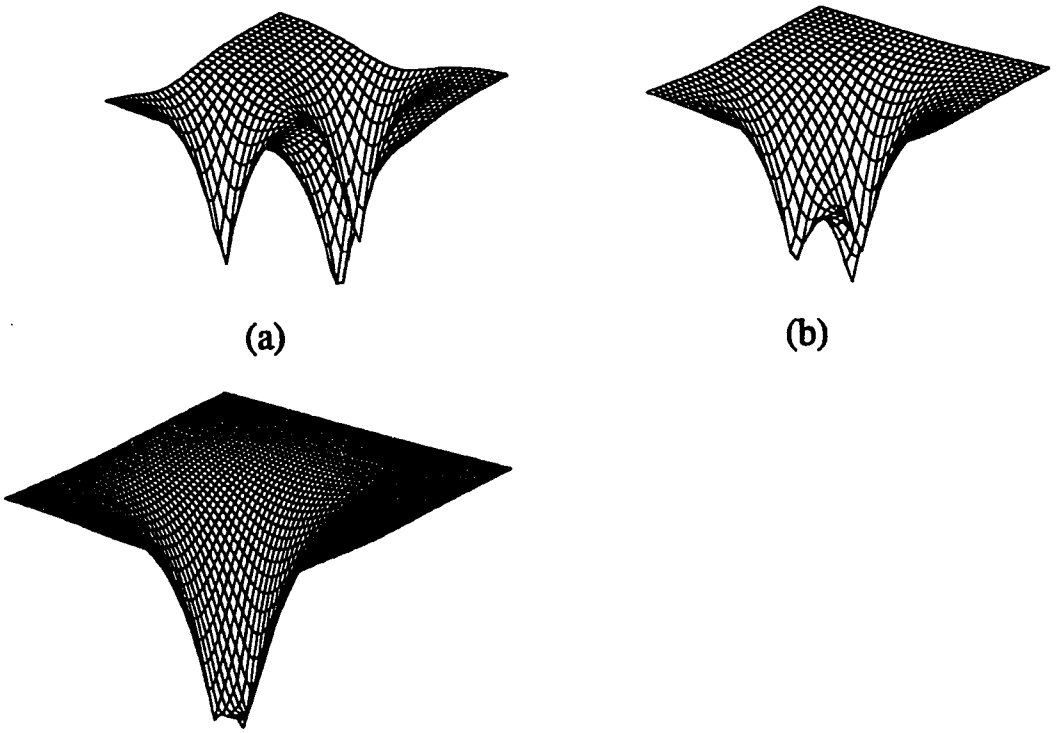


Figure 5.12. The sum of three symmetrically placed 2-d IDF's containing singularities. In (a)-(c) the functions are moved closer together. The same separation as in Figs. 5.11 are used. This figure may be compared directly with Fig 5.11 since the widths of the functions, as measured by second moments, have been set adjusted to be equal. Unlike the Gaussian IDF illustrated there, the local minima in the sum of the functions shown here *never* merge, even when they are very close together, as in (c). This shows that this type of function is completely unsuitable for use as an IDF.

Factor (c) in Table 5.1, that the IDF should be approximately parabolic at its minimum, insures that the CDF will result in an approximately least-squares fit of the predicted edge contours to the well matching, i.e., most similarly shaped, portions of the image edge contours. That is, AEFMA will return an approximate least squares fit to the pairs of shape primitives that dominate the CDF, i.e., those that match most closely. This is precisely the effect we desire to achieve. We can see that the multidimensional Gaussian of Eq. 5.2 has this property, since the Taylor series of Eq. 5.2 to second order yields

$$\mathrm{IDF}(\mathbf{p}^i, \mathbf{p}^p) \approx -1 + \sum_{k=1}^{N} \left[\frac{p_k^i - p_k^p}{\sigma_k} \right]^2 .$$

Since the $\sigma_k > 0$, $k = 0 \ldots N$, the multidimensional Gaussian IDF is parabolic in the neighborhood of 0.

Factor (d) in Table 5.1, that the IDF should approach zero for pairs of shape primitives that are very dissimilar, is crucial to successful minimization of the CDF. Pairs of shape primitives that are very dissimilar are not likely to be correctly matched. Therefore, the influence of such pairs should be deemphasized. In effect, since most of the primitive pairs are incorrectly matched, AEFMA can be viewed as fitting a model to data consisting mostly of outliers. Since this is the case, we must insure that the outliers do not distort the fit to the valid data. This is why we require that the IDF approach zero as the difference in the attributes becomes large. The multidimensional Gaussian satisfies this requirement, as Fig. 5.10 shows graphically for the 2-d case, since if any of the terms of the summation in Eq. 5.2 become large, then the argument of the exponential becomes a large negative number, resulting in an the IDF approaching zero.

Factor (e) in Table 5.1, that the IDF should smoothly blend the two extremes of the approximately parabolic region in the neighborhood of the minimum and the flat, nearly zero region far from the minimum, is partly a corrollary to the result that the IDF should have no curvature singularities. However, we desire the blending to be as smooth as possible. The reason is that it should be possible for a few erroneously matching pairs of primitives with very similar attributes to be overpowered by a large number

of correctly matching pairs of primitives with less similar attributes. If the transition occurs too quickly, then there is no chance that even a large number of weakly but correctly matching pairs of primitives will overpower a few accidentally, but strongly matching pairs of primitives. Of course, the transition is controlled by the values of the components of the σ-vector. However, viewed in normalized coordinates, where the effect of the σ-vector has been eliminated, the multidimensional Gaussian provides a very smooth transition from parabolic behavior near the minimum to flat, approximately zero behavior far from the minimum.

Factor (f) in Table 5.1, that the IDF should be parameterized to allow the size of the parabolic region to be easily adjusted, is accomplished with the multidimensional Gaussian by adjusting the size of the components of the σ-vector. We need to be able to do this in order to achieve AEFMA's coarse to fine behavior described in previous sections.

It is interesting to note that Siebert and Waxman [SW88] have proposed a method of grouping features together based on *diffusion* of the locations of the features that bears an interesting relation to the IDF and CDF. The problem they attempt to solve is to find the locations of significant groups of features at varying scales. The authors treat the locations of features, conceptually, as spots of ink in a 2-d water bath, and allow "time" to progress, resulting in the diffusion of the ink spots. As time progresses, the ink becomes more diffuse, and the local maxima in the concentrations of the ink disappear to be replaced by more global maxima. Siebert and Waxman propose to find significant groupings of features by looking for local maxima at different scales. It is well known that the solution of the diffusion equation under the conditions just described result in a summation of 2-d Gaussian distributions whose widths increase linearly with time, each centered at the location of the original ink spot, i.e., feature. Thus, while the methods otherwise have little in common, they do share the use of a summation of variable width Gaussians to smooth the contributions of features. In fact, Siebert and Waxman make

some of the same arguments as we do with respect to the fitness of Gaussians, particularly concerning the way that the summation of Gaussians in proximity tend to yield a single minima, not many local minima.

In this section, we have shown that multidimensional Gaussians of the form given in Eq. 5.2 satisfy all of our criteria with respect to the properties that the IDF should have in order to yield a CDF that will allow powerful continuous optimization procedures to quickly find the global minima with relative ease. We have not shown that multidimensional Gaussian are optimal for our task. Were we to attempt this, the criterion for optimality would rest on the propensity for individual minima in the function to merge into a single minimum in a summation, or to simply disappear. Indeed, it is likely that there are many possible artificially constructed functions that may work as well as multidimensional Gaussians. It is a topic of continuing research to define an optimality criterion that is suitable for this problem and find the function that achieves it. The author's intuition is that is will be a Gaussian. For now, however, we use Gaussians since they meet all the criteria described above, they have a simple analytic form, and finally, they are simple to compute.

5.3 2-d Shape Representation for Viewing Parameter Estimation

As we have seen, AEFMA is independent of any particular type of shape primitive, and any particular type of model used for predicting contours. This is advantageous as it allows the underlying framework to be easily adapted to other tasks that may be better suited by different shape representations or different types of models than those described here. This being said, we now describe the shape representation that we have used in AEFMA to estimate the viewing parameters of rigid objects, the focus of this thesis. It is the author's expectation, however, that this representation would be effective for many tasks so long as edge contours are available.

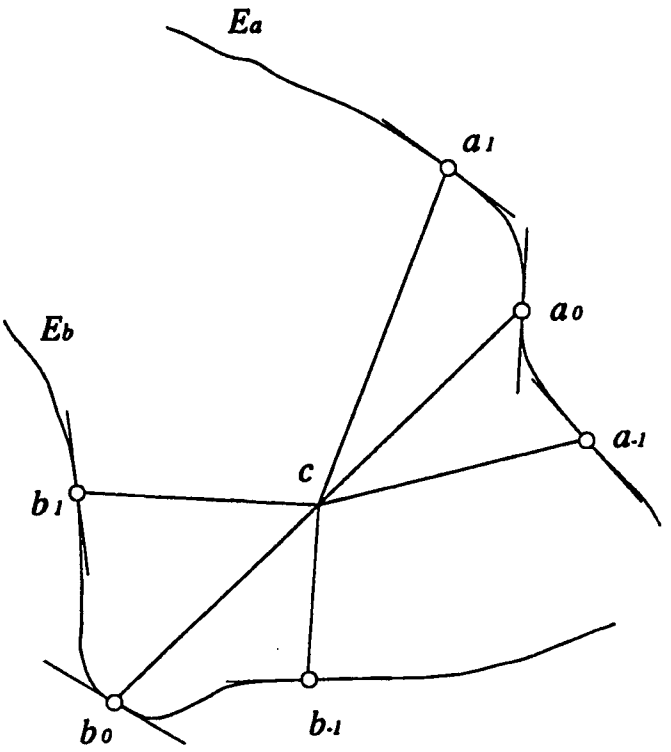


Figure 5.13. A shape primitive. The primary points of the primitive, a_0 and b_0, may be any two points on an edge contour. The auxiliary points, $a_{\pm 1}$ and $b_{\pm 1}$, are derived by traversing a length of edge contour, αl, starting from a_0 or b_0, where l is the length of the line segment $a_0 b_0$, and α is a proportionality constant. For each pair of points, several shape attributes are calculated: the relative angles between the tangents at a_n and b_n, $n = 0, \pm 1$; the normalized lengths of ca_n and cb_n Section 4.3.2, $n = \pm 1$, where $l = d(ca_0)$; and the relative orientations of ca_n and cb_n, $n = \pm 1$, relative to $a_0 b_0$.

The abstract "shape primitives" that we have referred to throughout the preceding discussion are, in fact, very similar to the point-pair features described in Section 4.3.2. Indeed, the only difference is that the primary points of the features described in Section 4.3.2 were required to be either inflection points or critical points of the edge contour. Here, on the other hand, the primary points may be any two edge points. Fig. 5.13, shows a shape primitive.

The attributes of the shape primitives are identical to the attributes of the point pair

feature, i.e, referring to Fig. 5.13, the primitive is derived from two edge contours E_a and E_b, and points a_0 and b_0 are the primary points of the primitive. Point c is the center of line a_0b_0. If l is the length of a_0c (or b_0c), then points a_1 and a_2 are located a distance αl on either side of a_0 along the curve E_a, where α controls the degree of spatial locality of the shape primitives. The *invariant attributes*, i.e., the attributes that do not change if the feature is translated, rotated, or scaled within the image plane, include:

- the normalized distances $d_n^a \equiv d(c, a_n)/l$, $n = \pm 1$, and where $d(c, a_n)$ is the distance between points c and a_n;
- the normalized distances $d_n^b \equiv d(c, b_n)/l$, $n = \pm 1$, and where $d(c, b_n)$ is the distance between points c and b_n;
- the orientations o_n^a of the segments ca_n with respect to segment ca_0, $n = \pm 1$;
- the orientations o_n^b of the segments cb_n with respect to segment cb_0, $n = \pm 1$;
- the angles t_n^a between line ca_0 and the tangent to E_a at a_n, $n = 0, \pm 1$;
- the angles t_n^b between line cb_0 and the tangent to E_b at b_n, $n = 0, \pm 1$;

while the *variant attributes*, i.e., those that vary when the primitive is translated, rotated, or scaled in the image plane include:

- the x-y coordinates of c;
- the orientation of a_0b_0;
- the length l of a_0b_0.

Section 4.3.2 shows that the invariant attributes of the features are indeed invariant to image plane translation, rotation, and scaling.

Section 4.3 discussed a number of properties of good features for initial hypothesis generation. A subset of these properties are also important for our shape primitives to possess. As before, the following properties are useful to varying degrees:

- spatial locality,
- selectiveness,
- noise resistance,

- insensitivity or invariance with respect to variations in insignificant viewing parameters,
- smoothness with respect to variations in significant viewing parameters,
- ease of segmentation, and
- representational compactness.

Most of these properties, as discussed in Chapter 4, apply to shape primitives in the same manner as features. However three of these properties, specifically, selectiveness, smoothness, and invariance, have somewhat different significance with respect to shape primitives than with respect to the point-pair features used by the hypothesis generation process. In particular, selectiveness, and smoothness remain important properties On the other hand, invariance, while convenient, is no longer essential. We now examine the reasons for this.

5.3.1 Selectiveness in the Context of Shape primitives

Selectiveness is a very useful property for shape primitives to have because the IDF can then be neglected for all but a few well-matching primitive pairs. This permits primitives with well-matching shape attributes to overcome the effects of proximity attributes. This, in turn, is the essential reason that AEFMA can converge to the correct object and pose *even from distant poses and through clutter and obstacles*. Experimentally, we show that AEFMA is able to converge to the correct viewing parameters even when the initial guess is very poor, and when there is clutter and occlusion. This behavior is the direct result of using selective shape primitives.

Fig. 5.14, Fig. 5.15, and Fig. 5.16 all show a model contour, drawn in gray, superimposed on edge contours detected in an image, drawn in black. In all cases, a predicted shape primitive has been selected from the set of all predicted shape primitives. This primitive is drawn in black. Shown also are the image primitives that result in a significant value of the IDF when paired with the predicted primitive. The darkness of the

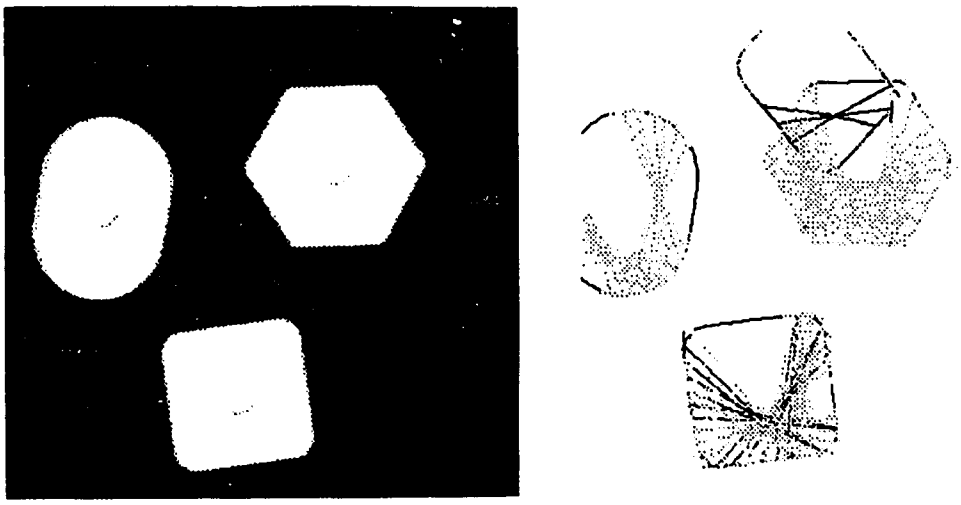
primitive corresponds to the how close the IDF is to its minimum of -1. Solid black indicates an $|IDF| \approx 1$, while white indicates an $|IDF| < 10^{-5}$, with a logarithmic variation between these extremes. The figures show that, even when the predicted contour is distant from the correctly matching image contour, only the strongly contributing image primitives on a correctly matching part of the image contour have large values of |IDF|. In this example, the components of the σ-vector were set to larger values, with the result that the IDF is less sensitive to location. This configuration of the σ-vector is typical for the early stages of AEFMA.

The selectivity of these primitives demonstrated in Figs. 5.14, 5.15, and 5.16 holds for the vast majority of primitives on the predicted contour. Thus, the optimization procedure will tend to vary the viewing parameters in order to improve the match with the "dark", i.e., most significant, image features. This almost invariably results in an improvement in the viewing parameters. However, as described earlier, in order to get greater accuracy in the estimates of the viewing parameters, the σ-vector's components must be reduced in value, and the optimization procedure rerun.

While selective shape primitives are useful for improving AEFMA's range of convergence, AEFMA can function reasonably well without selective primitives. The reduced selectiveness of the shape primitives translates directly into a reduction in AEFMA's range of convergence. Section 5.9 will show some examples of a CDF that is very similar to the local, image gradient-based methods used in methods employing deformable models [KWT87, ATW88, TWK36, AWJed]. This type of CDF works quite well so long as the initial viewing parameters are fairly accurate.

5.3.2 Attribute Smoothness in the Context of Shape primitives

That the attributes of the primitives vary smoothly with respect to the viewing parameters is of critical importance to AEFMA. If the attributes do not change smoothly, singularities will be introduced into the IDF. As we have seen, this leads to the creation



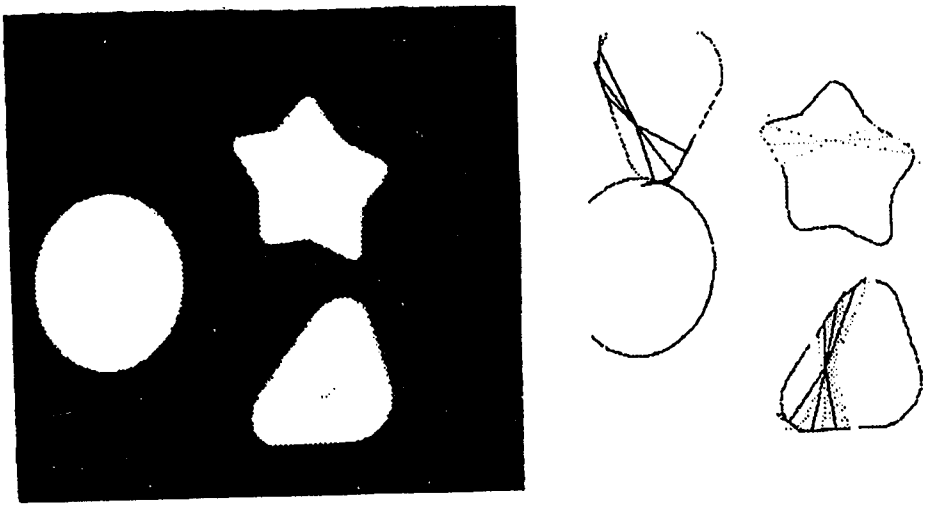
(a) (b)

Figure 5.14. An empirical demonstration of the selectiveness of AEFMA's shape primitives. An image of smooth geometric objects is shown in (a). In (b) the edge contours that were detected are drawn in black, while the predicted contour is shown in gray. A shape primitive, drawn in black, has been selected at random from among the shape primitives comprising the shape representation of the predicted edge contour. Drawn in varying shades are the shape primitives comprising the image shape representation. The blackness of the image shape primitive indicates the magnitude of the IDF: black indicates most similar, i.e., $|IDF| = 1$, while white indicates $|IDF| < 10^{-5}$, with a logarithmic variation between these extremes.

of spurious local minima that will trap the optimization procedure and lead to erroneous results. The attributes of the shape primitives we have been describing do have the property that they vary smoothly over wide regions of viewing parameter space.

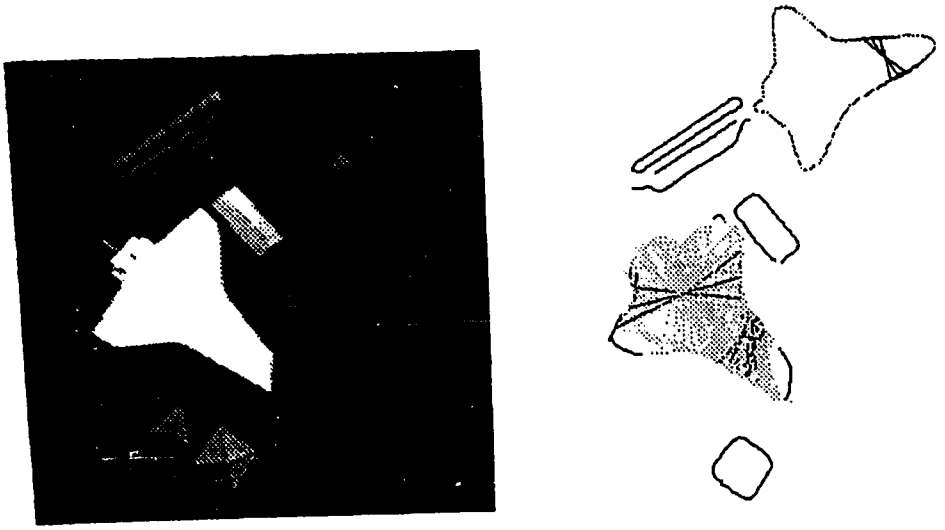
5.3.3 Attribute Invariance in the Context of Shape primitives

Invariance of the attributes of AEFMA's shape attributes is not as important as invariance of the features was to the hypothesis generation process. Since the optimization



(a) (b)

Figure 5.15. This figure is analogous to Fig. 5.14.



(a) (b)

Figure 5.16. This figure is analogous to Fig. 5.14.

procedure is operating withing the 6-d space of viewing parameters, and, generally, all of the attributes of a shape primitive will vary when an out-of-plane rotation occurs, it hardly matters if the attributes change with respect to other viewing paramters. However, having as many invariant attributes as possible is convenient since it allows the effects changing different components of the σ-vector to be decoupled from one or more of the

viewing parameters. For example, since the coordinates of a predicted shape primitive's center are the only attributes that change when the model is translated parallel to the image plane, the width of the CDF in these directions is controlled only by the size of σ_x and σ_y, the corresponding components of the sigma vector. There is some coupling, however, due to the aggregate nature of the CDF. For example, changing σ_x and σ_y may affect the CDF along the dimension of the scale parameters, since changing the scale of an object will generally change the locations of the centers of the primitives. Nevertheless, invariance provides decoupling that simplifies the interactions that may occur.

5.3.4 Computation of the Shape primitives

Computation of shape primitives is identical in most respects to the computation of the point-pair features described in Section 4.3.2.6. Please refer there for the complete discussion. Here we will mention a few issues that are specific to the shape primitives.

The essential difference between a shape primitive and the point-pair features of Section 4.3.2.6 is that each of the primary points of a point-pair feature are required to be either an inflection point or a critical point whereas the primary points of a shape primitive may be any edge contour point. Thus, the computation of shape primitives is simpler than point-pair features in that inflection points and critical points need not be detected.

As mentioned in Section 4.3.2.6, there is a problem with the symmetry that exists among the attributes of the shape primitives. In contrast to the point-pair features of Section 4.3.2.6 where, when one of the primary points is an inflection point and the other is a critical point, the primary points of the shape primitives cannot be distinguished *a priori*. This is analogous to the case when both of the primary points of a point-pair feature were of the same type, i.e., both were either inflection points or critical points. Thus, referring to Fig. 4.21, which has been duplicated in Fig. 5.17 for convenience, there are eight possible shape primitives that could result from identical edge structures.

This would require that, in order to compute the IDF, eight permutations of the feature attributes would have to be used. Aside from being eight times as much work, there would be a larger chance of erroneously matching features with a large values of the IDF distorting the CDF. The solution is the same as that described in Section 4.3.2.6: put the primitives into a canonical form that allows the vast majority of them to be compared directly, attribute by attribute, thus cutting the work that must be done by roughly a factor of eight, and improving the nature of the CDF by eliminating many meaningless terms in its computation.

5.3.5 Sparse versus Complete Shape Representation

We have described the shape primitives used by AEFMA to generate the CDF. As yet, we have not described the number or placement of the shape primitives in the overall shape representation. Our experience has led us to use a dense placement of the shape primitives over the image contours, with a sparse placement of primitives over the predicted contours.

When AEFMA was originally being designed, we had hoped to be able to use a sparse shape representation for both the observed and the predicted edge contours. This is possible only if the primary points of the shape primitives can be accurately aligned using marker points, such as critical points or inflection points. To see why, suppose, for example, that the primary points, are spread evenly over the edge contours. As shown in Fig. 5.18, when the primary points of the shape primitives are sparsely spaced, it is unlikely that any of the predicted shape primitives are likely to be describing a portion of contour that is also being described by an image shape primitive. Thus, none of the predicted primitives will have well-matching image primitives. In such a scenario, the CDF will have minima whose localization is limited by the density of the spacing of the image primitives.

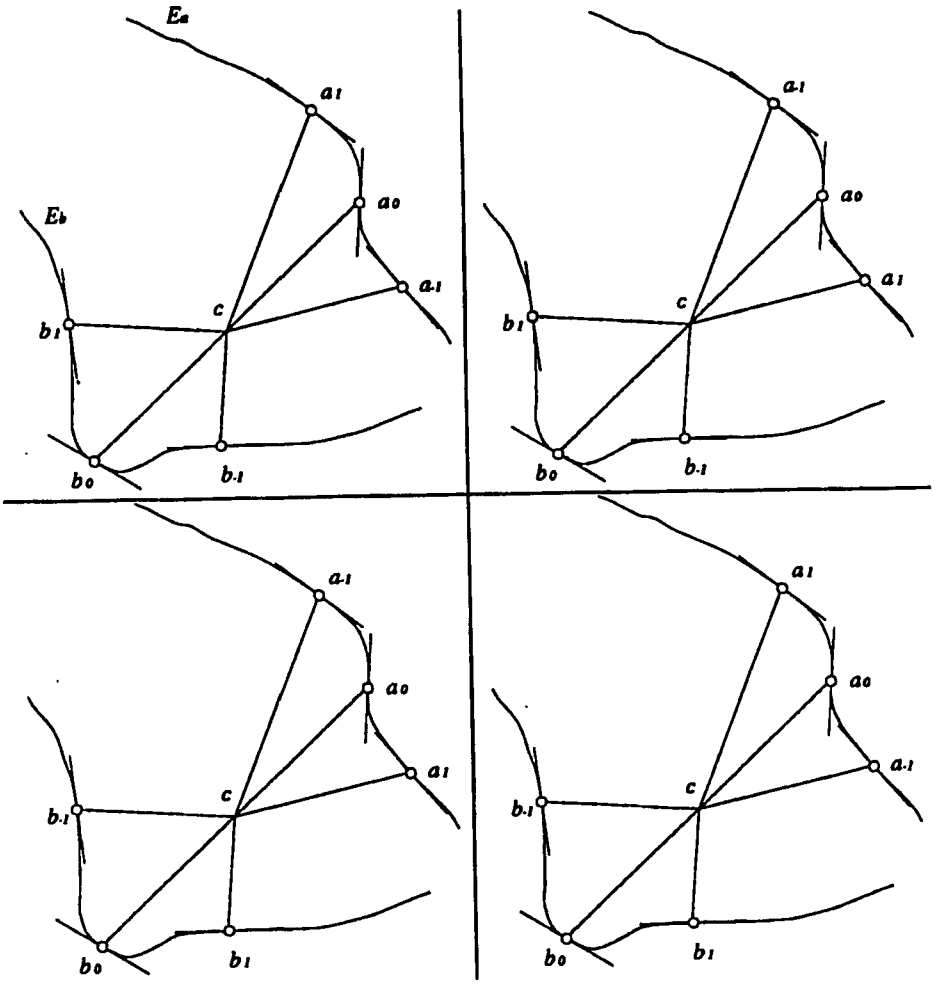


Figure 5.17. The figure shows the four possible assignments of the auxiliary points a_{-1}, a_1, b_{-1}, and b_1. Four other assignments occur when a_0 and b_0 are switched. Which assignment is actually realized depends on the direction of the parameterization of the edge contours and the order in which the edge contours were detected. Each of the possible assignments of the auxiliary points leads to shape primitives that are generally distant from each other in primitive-attribute space, in spite of the fact that they possess visually identical structures. We wish to avoid this situation as it defeats the principle that primitives that are similar in appearance should also possess similar attributes.

If marker points are used, the situation can be improved since, as the model is more accurately aligned to the image, the marker points, as shown in Fig. 5.19 will be located

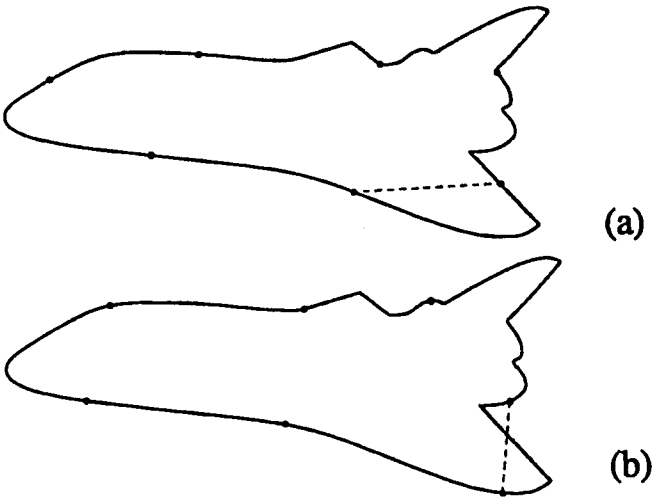


Figure 5.18. If the primary points of the shape primitives are sparsely, but evenly, spaced over both the predicted edge contour (a) and the image edge contour (b), then, due to starting point ambiguities, breaks in the image contour, and other phenomena, it is unlikely that in the placement of primary points on the predicted edge contour will match the placement of primary points on the image edge contour. Thus, even when the contours are perfectly matched, as in (a) and (b), there will be few or no image shape primitives that match well to any predicted shape primitives. An example is shown in the figure of a predicted shape primitive and the nearest image shape primitive. The result is that the CDF is unlikely to have a well defined global minimum. Rather, several strong, incorrect local minima are likely to exist, making it unlikely to find the correct viewing paramters.

on accurately corresponding portions of the predicted and observed contours. If this is so, why not use this approach? Unfortunately, there is a major drawback: marker points may appear and disappear from the predicted contour in unpredictable ways. Recall that AEFMA forms no hard correspondences between the 3-d model and the image contours. Therefore, the only way to determine whether marker points exist in the predicted contour is to apply a detection algorithm that is essentially identical to the one used to detect such points in the image contours. Thus, as the predicted contours alter their shape as the viewing parameters are varied, the marker points will appear and disappear depending

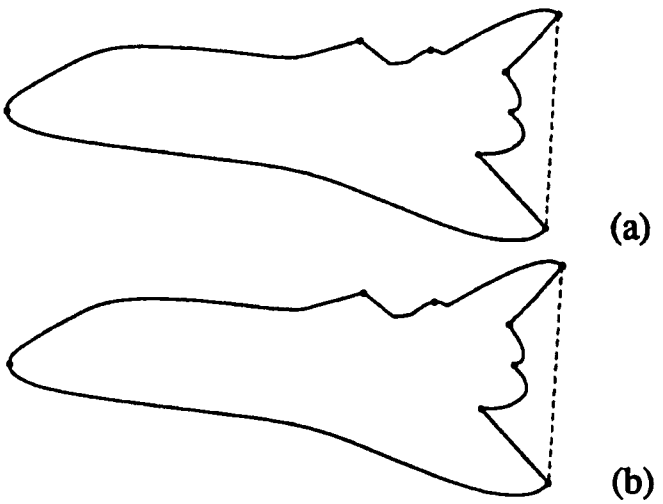


Figure 5.19. Again, (a) denotes the predicted edge contour and (b) denotes the detected edge contour. If certain points, called marker points, can be reliably found on the edge contours, then the primary points of the shape primitives may be sparsely placed since the marker points will enforce proper alignment, as shown above. Candidates for such points are critical points and inflection points of the contours. Unfortunately, since the predicted edge contour's shape is varying as the viewing parameters are adjusted in the course of the algorithm, the marker points may appear and disappear as detection thresholds are crossed and recrossed. As discussed in the text, this leads to discontinuities in the CDF, making it difficult to minimize.

on the various thresholds and parameters of the marker point detection algorithm. Such unpredictable appearances and disappearances of marker points wreak havoc with the CDF, since terms are appearing and disappearing.

In essence, what is being described in the previous paragraph is the result of the limit of the summation in Eq. 5.1, N, becoming a function of the viewing parameters. N is an integer, therefore it changes value abruptly, and the result is discontinuities in the CDF. The effect of the discontinuities can be reduced if Eq. 5.1 is premultiplied by a factor of $1/N$. In this case, if a marker point disappears from one set of viewing parameters to another, and if the IDF values corresponding to the lost pairs of primitives

"average" to the same value as the CDF's previous value, then the CDF will retain the same value, avoiding the creation of a discontinuity. However, if the values of the IDF's corresponding to the lost pairs of primitives tend to be larger than the CDF, then the value of the CDF would have a positive discontinuity. The reverse situation would result in a negative discontinuity. Unfortunately, the "average" of the IDF values of the lost or gained pairs of primitives is rarely, or never, the same as the original value of the CDF. Thus, the CDF becomes discontinuous, making it difficult or impossible to minimize. Further, the sparsity of the representation still leads to more pronounced local minima in the CDF.

The discontinuities introduced by the use of marker points result in artificial barriers, such as that shown in Fig. 5.20, noise, and spurious local minima in the CDF. Fig. 5.21 shows a 2-d slice of the viewing parameters (the 2-d space of out of plane rotations), of a CDF using critical points as marker points. As can be seen, this function would be difficult for any optimization procedure, continuous or discrete, to minimize. Due to this fundamental problem, the marker point approach must be scrapped.

Fortunately, the problems of the marker point approach to shape representation can be overcome. Our solution is to *densely*, but, evenly space the primitives' primary points on the image edge contours, and sparsely space them on the predicted edge contours. Since the image shape representation needs to be calculated once for each image, whereas the predicted shape representation may be computed many times, this imbalance is justified from a computational standpoint. More importantly, this approach results in a well behaved CDF. Since the image contours are represented densely by primitives built from evenly spaced primary points, the problems of accuracy resulting from sparse placement of image primitives in the absence of marker points are avoided. Given that the image edge contours are densely covered with primary points, the predicted edge contour may be sparsely covered for the following reason. The dense coverage of the image edge contours implies that it is likely that a shape primitive derived from any points on the

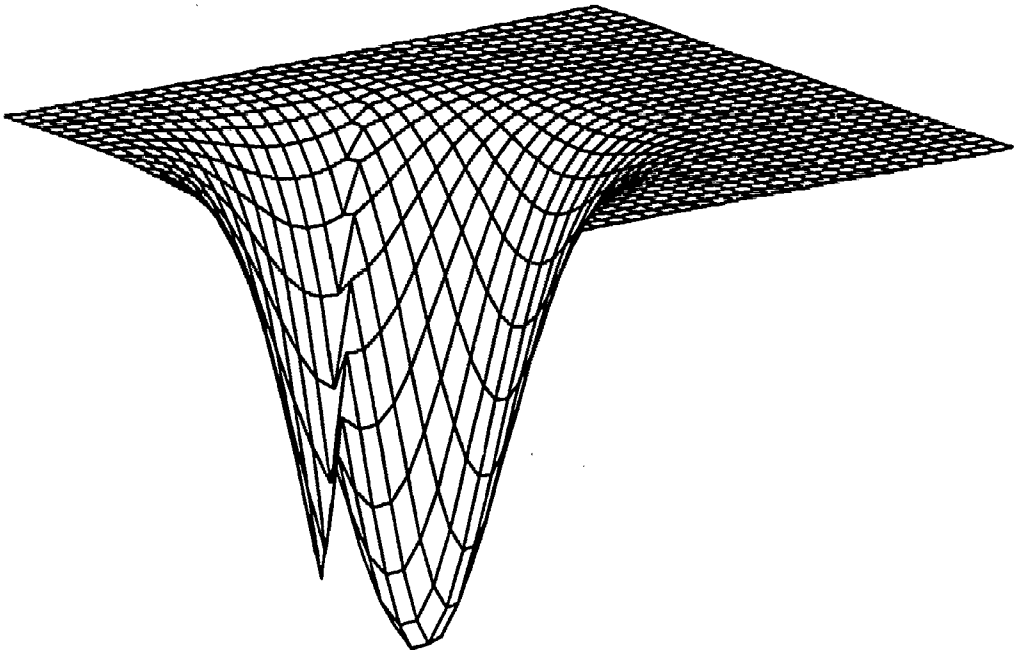


Figure 5.20. This figure shows a 2-d slice of a hypothetical CDF produced when marker points are used. In the space of viewing parameter, when the barrier is crossed, a marker point on the predicted contour appears or disappears. When that occurs, the number of terms in the summation in Eq. 5.1, N, may change drastically, almost always resulting in a discontinuity in the CDF.

predicted edge contour will have a corresponding image shape primitive that describes a correctly matching portion of the image edge contour, if it is visible or undistorted in the image. Thus, the minima of the CDF will occur where the shapes of the predicted and observed contours are really the most similar, not where the predicted shape primitives accidentally match one of a few image primitives. Later, in Section 5.5.2, we show 2-d slices of the well conditioned CDF's that result from the approach of densely covering the image edge contours and sparsely covering the predicted edge contours. As can be seen there, the resulting CDF's are smooth, with a clearly defined global minimum with no spurious local minima nearby. This is the approach we use in AEFMA.

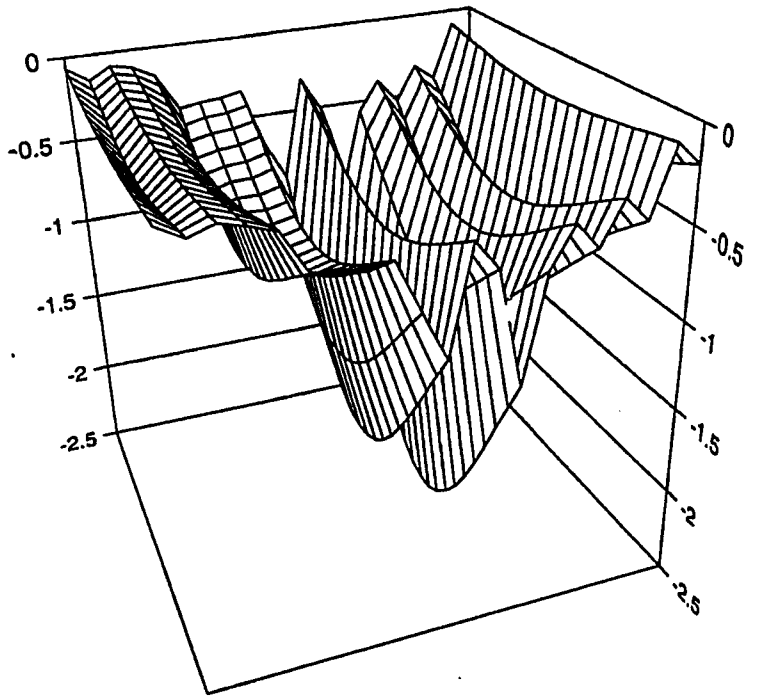


Figure 5.21. This figure shows a 2-d slice of an actual CDF produced when marker points are used. As can be seen, many spurious local minima and barriers are produced. The minimization techniques we used (see Section 5.5.3) failed to find the global minimum of this function.

5.3.6 Computation of the Shape Representation

Given a set of N primary points that are evenly spaced over a set of edge contours, the complexity of forming primitives from all possible pairs of primary points is $O(N^2)$. Thus, the cost of densely spacing the primary points over the image contours could be substantial if all possible pairs of image features were computed. Fortunately, it is not necessary, or even desirable, to compute all possible primitives. There are two mechanisms by which the complexity can be reduced. The first mechanism is to use information from the contour grouping module, and possibly from the incremental verification process, to form primitives from pairs of primary points that are likely to be from the same object. The second mechanism is to *not* form primitives from points that are on the same

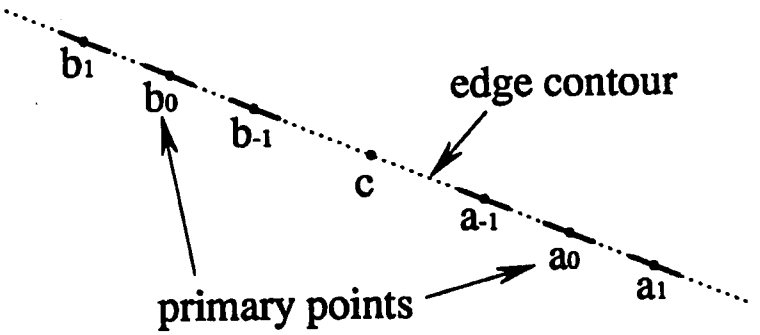


Figure 5.22. When the primary points, a_0 and b_0 are on the same edge contour, and they are close together, the primitives tend to all look like the one shown above. This is due to the fact that the edge contours are approximately linear at a small enough scale, and that the description of a line always the same.

contour, *and* are too close together. The reason for this is that primitives formed from such points are unselective because edge contours tend to look alike, i.e., linear, when viewed from a small enough scale, as shown in Fig 5.22. These measures reduce the number of points markedly. As a typical example, without the above measures, 11,314 primitives were computed from the image, whereas putting them into effect resulted in only 1,018 primitives.

The considerations of the previous paragraph are less important for the predicted shape representation since it consists of many fewer primary points. For example, the image shape representation may contain 200 primary points, whereas the predicted shape representation may contain as few as five, but never more than nine. Nevertheless, empirically, we have found that excluding the primitives consisting of primary points that are close together lessens the number and severity of the local minima in the CDF, resulting in further improvements to AEFMA's range of convergence. However, the filtering must be done carefully, otherwise the number of primitives may fluctuate when the size of the predicted contour changes, potentially leading to the same problem encountered with the marker point approach mentioned in Section 5.3.5. The following paragraph explains how this is done.

Image primitives are proximity filtered by computing the distance between the primary

points (the length of line segment a_0b_0 in Fig 5.13), and computing the attributes of the feature *only* if the distance is larger than a threshold T_p. In order to avoid the problem brought up in the previous paragraph, we modify this simple approach for the predicted contours by multiplying the threshold provided by the user, T_p^u, by the scale parameter of the model, s, to obtain T_p from $T_p = sT_p^u$. Thus, when the scale parameter of the model is changed, the distance between the primary points of the primitives changes, but the proximity threshold T_p changes by exactly the same factor, resulting in the computation of exactly the same primitives as the unscaled contour, thus avoiding the problem of appearing and disappearing primitives encountered by the marker point approach.

5.4 Computing the Composite Disparity Function

Part of the innovation of AEFMA is a way to compute the composite disparity function very efficiently. As is often true, necessity was the motivation, since the method would be impractical were it not for speedups described in this section.

5.4.1 Computing the Composite Disparity Function Efficiently

Suppose that the predicted shape representation consists of n^p primitives and the image shape representation consists of n^i primitives. Then, there will be $n^i n^p$ terms in the summation in Eq. 5.1. As mentioned earlier, it is not unusual to have more than 1,000 image primitives. A typical number of predicted primitives is 36. In this situation, there could easily be more than 50,000 IDF terms in Eq. 5.1, rendering the computation of the CDF very slow if all of these terms actually needed to be computed. Fortunately, most of the terms of Eq. 5.1 are negligible, permitting the vast majority of them to be ignored and only the significant terms in the summation to be computed. For example, if there are an average of four significant contributors to the summation for each predicted shape primitive (the actual number depends on the size of the components of the σ-vector, but

four is reasonable), the total number of IDF terms that need to actually be computed is reduced from 50,000 to 200, a much more manageable number.

Why are most of the terms of Eq. 5.1 negligible? Referring to the defining equation of the IDF, Eq. 5.2, we see that in order for the IDF($\mathbf{p}^i, \mathbf{p}^p$) to be near its minimum of -1, the components of the predicted shape primitive vector, $\mathbf{p}^p_k$, $k = 1 \ldots N$, and the components of the image shape primitive, $\mathbf{p}^i_k$, $k = 1 \ldots N$, must be near each other. If they are not, the IDF will be close to zero. We have found empirically that requiring $|\mathbf{p}^p_k - \mathbf{p}^i_k| <\approx 2\sigma_k$ works well, where, as before, N is the dimension of the primitives' attribute vector, and σ_k is the k^{th} component of the σ-vector. The probability that all of the attributes of an incorrectly matching pair of primitives will fall into such a range by chance is very small. For this reason, most of the terms of the summation will be negligible.

We have argued theoretically, and observed empirically, that the number of non-negligible IDF terms in the defining equation of the CDF is far smaller than the total number of terms. The question remains as to how to efficiently determine which of the terms is actually significant without computing them all. The answer lies in the observation that, given a predicted shape primitive, then the image shape primitives that will have significant IDF's when paired with this predicted primitive are within a *neighborhood*, whose size is determined by the σ-vector of the predicted primitive, in primitive-attribute space. Thus, we may find all of the image primitives that may contribute significantly to the CDF when paired with a particular predicted primitive by performing a neighborhood query on the set of image shape primitives using the a neighborhood defined by

$$N(\mathbf{p}^p, \sigma) = \{\mathbf{x} \mid p^p_k - \beta\sigma_k < x_k < p^p_k + \beta\sigma_k, k = 1 \ldots N\}, \tag{5.3}$$

where β is a factor that controls the size of the neighborhood relative to the σ_k. We have commonly used $\beta = 2.0$.

Recalling the discussion of neighborhood queries in Section 4.2.3, we can perform

such queries in time proportional to the logarithm of the size of the set of image shape primitives using k-d trees. Thus, the **detected_shape_database** is actually a k-d tree, and the function **get_significant_contributors** performs neighborhood query using the k-d tree to retrieve, for each predicted shape primitive, the image shape primitives that will result in a non-negligible IDF when paired with the predicted primitive. Refer to Section 4.2.3 for details on k-d trees and how to use them for neighborhood queries. As shown earlier, the IDF needs to be computed *only* for this subset of pairs of primitives. Since the number of such pairs is usually 2-3 orders of magnitude smaller than the total number possible, a great speedup is realized.

5.4.2 Weighting the Shape primitives

In previous sections, we have stressed the importance of not allowing predicted shape primitives to simply appear or disappear as the viewing parameters vary. There is an interesting phenomenon that occurs near the endpoints of contours which mimics this behavior. Fortunately, overcoming this problem is relatively simple. Its roots lay with the fact that primitives that have an auxiliary point that falls off the end of a contour is rejected as invalid. Clearly this is necessary, as many of the attributes of such a feature are meaningless. The problem is that, a small change in the viewing parameters can result in a predicted shape primitive going from being valid to being rejected as invalid. This causes precisely the types of problems described in Section 5.3.5, because, when valid, the primitive may be contributing to the CDF, and when it ceases to exist, a discontinuity results in the CDF.

We have addressed this problem by smoothly "phasing out" any primitives that possess any points, auxiliary or primary, that are too close to the endpoints of the contour. This works as follows: any shape primitive that has all of its component points farther than a distance S, in arclength, from any of the endpoints of its component contours is given full weighting. Any primitive possessing a point that is closer than S to any endpoint is

given a lower weight. The reduction is calculated to smoothly blend from full weighting to zero weight as a function of the distance that the nearest point of the primitive to an endpoint. Many functions could be used; we use a shifted half-Gaussian. Thus, a primitive that begins to approach the end of one of its parent contours causes the value of the IDF of any pair of primitives in which it is participating to be reduced smoothly. If it finally disappears, it is of no consequence since, by that time, its contribution would have been negligible in any case. Thus, the smoothness and continuity of the CDF is preserved.

While not as important, we weight the image primitives in the same manner. This reduces the effect of image primitives that have unreliable attributes from accidentally matching well to some predicted shape primitive. The probability of such an occurrence is rather small, and, indeed, there is not a noticeable change in performance when this feature is turned off and on.

5.5 Minimizing the Composite Disparity Function

We turn now to the nature of the CDF and the best approach to minimizing it as a function of the viewing parameters.

The viewing parameters provide a parametric description of the transformation between the object coordinate frame and viewer, or camera, coordinate frame. That is, in the language of Section 1.2.1, the viewing parameters provide a parametric description of $\mathcal{T}_{co}$, the transformation mapping a point in the model coordinate frame to the camera's coordinate frame. There are many possible parametric description of $\mathcal{T}_{co}$, and the choice is largely a matter of personal preference. We have chosen a parameterization in AEFMA that separates the image plane transformations, i.e., translation, scaling, and image-plane rotation, from the out of plane rotation. We do this because it allows us to take advantage of the decoupling of the parameters, discussed previously in Section 5.3, that this engenders.

The rest of this chapter will occasionally need to refer to specific viewing parameters. Therefore, we now describe the particular parametric description of $\mathcal{T}_{co}$ that we are using within AEFMA.

5.5.1 Viewpoint Parameterization

The parametric description of $\mathcal{T}_{co}$ will be in terms of a sequence of coordinate transformations applied to the camera coordinate frame, assuming that the object and camera frames are initially aligned. As described in Section 1.2.2, the image plane is perpendicular to the z axis in the camera's coordinate plane. Weak perspective, described in Section 1.2.2, will be assumed.

The first two transformations parameterize changes in viewpoint that cause the object to appear to rotate out of the image plane, i.e., the rotation *axis* is confined to the image plane. The first transformation is a positive rotation of α about the object frame's x axis[1]. By definition, a positive rotation about an axis follows the right-hand-rule. The rotation about the object frame's x axis is followed by a rotation about the object frame's z axis. Thus, α and β parameterize the viewing directions, shown geometrically in Fig. 5.23. Essentially, the view directions are parameterized by the "latitude", α, and the "longitude", β.

The third transformation parameterizes image-plane rotations that do not affect the shape of the objects appearing in the scene. It is characterized by a rotation of θ about the camera frame's z axis. This effect is observed when you rotate a camera without moving the direction in which it is pointing.

The fourth transformation, under weak perspective, parameterizes the scale change that occurs as the camera frame becomes more distant from the object frame. The transformation is a translation of r in the $-z$ direction of the camera frame. As shown

---

[1] At this point, since the camera and object frames are aligned, a negative rotation of α could equally well be made about the camera frame's axis.

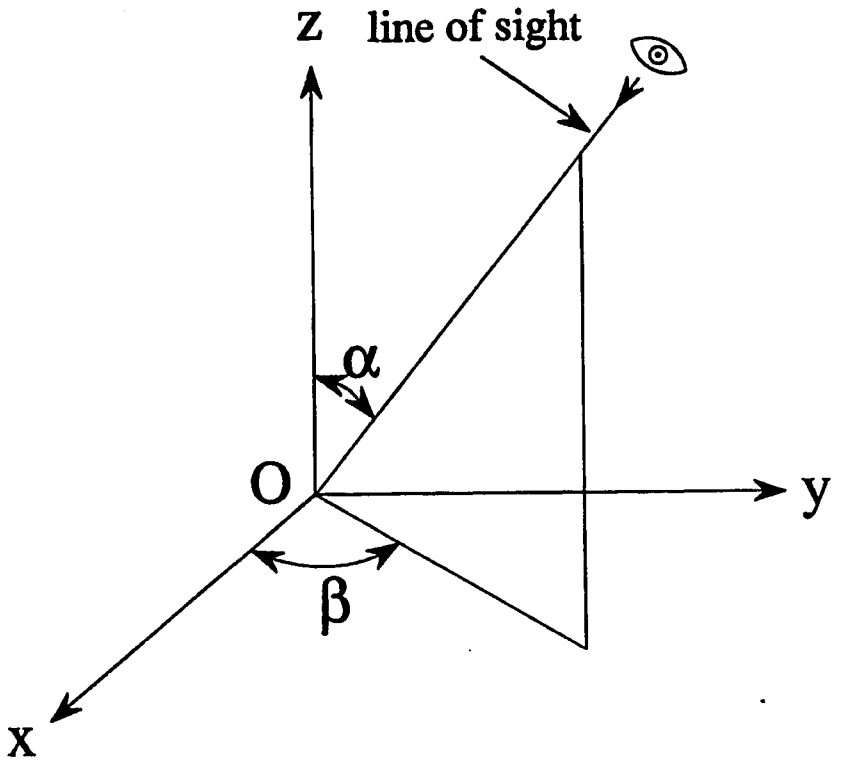


Figure 5.23. The parameterization of the viewing parameters that result in rotations of the scene out of the image plane is shown in terms of the angles α and β. The point O is the origin of the object coordinate system.

in Eq. 1.2, under weak perspective, this amounts to a scale change $s = f/r$, where f is the focal length of the camera. Under full perspective, we would use r to parameterize the transformation. However, under weak perspective, it is convenient to use s as the parameter as it simplifies calculations. Varying s is like zooming in or out with a zoom lens.

The final transformations are simple translations parallel to the image plane. It can be shown that panning a camera closely approximates this transformation when the camera is distant from the object. This transformation is parametrically described by a translation of x along the camera frame's x axis and by a translation of y along its y axis.

The homogeneous matrix representation of $\mathcal{T}_{co}$, $\mathbf{T}_{co}$, is then

$$\mathbf{T}_{co} = \begin{bmatrix} c\theta c\beta - s\theta c\alpha s\beta & -c\theta s\beta - s\theta c\alpha c\beta & s\alpha s\theta & x/s \\ s\theta c\beta + c\theta c\alpha s\beta & c\theta c\alpha c\beta - s\beta s\theta & -s\alpha c\theta & y/s \\ s\alpha s\beta & s\alpha c\beta & c\alpha & 0 \\ 0 & 0 & 00 & 1/s \end{bmatrix}, \tag{5.4}$$

where $c\alpha \equiv \cos(\alpha)$, $s\alpha \equiv \sin(\alpha)$, $c\beta \equiv \cos(\beta)$, $s\beta \equiv \sin(\beta)$, $c\theta \equiv \cos(\theta)$, and $s\theta \equiv \sin(\theta)$. The homogeneous image coordinates, $\mathbf{u} = (u_x, u_y)$, of a homogeneous 3-d point $\mathbf{r} = (x, y, z, 1)^t$ are obtained from the equation

$$\mathbf{u} = \mathbf{P}\mathbf{T}_{co}\mathbf{r}, \tag{5.5}$$

where P is a 3×4 projection matrix

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

5.5.2 Empirical Observations on the Nature of the CDF

Throughout this chapter we have attempted to make convincing arguments that the CDF will be a easy function to minimize, and, therefore, will be easily minimized by well known continuous optimization procedures. The purpose of this section is to show empirically that these arguments are, indeed, borne out.

In the following, we will provide 3-d plots of the CDF as a function of two of the viewing parameters. While divining the true nature of the CDF in the 6-d parameter space from 2-d slices is not trivial, these plots provide reasonable insight into the behavior of the CDF.

Fig. 5.24 shows a plot of the CDF versus the translational viewing parameters x and y (see the definitions of the viewing parameters in Section 5.5.1). The center of the

plot represents the minimum of the CDF. The other viewing parameters have also been adjusted in this figure to the global minimum of the CDF. In the plot, the limits of the plot are $\pm.2$ from the central values of x and y (in this case, $x = .122$ and $y = -.077$, though this is unimportant). We will refer to this as dx and dy. Thus, $dx = dy = .2$. A 21×21 grid of the values of the CDF was taken. Figs. 5.25 (a), (b), and (c) show the translational bounds of the parameters. As can be seen, the CDF has a single, strong minimum at the bottom of a smooth, circular "valley". Outside the confines of the valley, small undulations can be seen. Virtually any continuous minimization method would converge to the correct minimum of this function, so long as the starting point was within the "valley". The size of the "valley", and therefore the range of reliable convergence to the minimum, is influenced in this case primarily by the σ_x and σ_y components of the σ-vector. In this case, $\sigma_x = \sigma_y = .1$ (recall that the translational bounds of the image are $x, y = \pm 1$).

Figs. 5.26 and 5.27 are analogous to Fig. 5.24 in all respects except that the values of σ_x and σ_y have been reduced to .05 and .025 respectively. The improved localization of the minima as well as the reduced size of the "valleys" can be seen. Note that the size of the valley tends to be roughly proportional to σ_x and σ_y. This tends to hold for other components of the σ-vector as well.

As the above plots have shown, the CDF is a well behaved function with weak local minima outside of a primary "valley" whose bottom is the global minimum. Most continuous optimization methods could easily and quickly locate the minimum of the CDF if they were supplied with a starting point that was within the confines of the "valleys" of the above figures. In the Cyclops framework, getting the initial estimate to fall within the "valley" is the job of the hypothesis generator. In a model-based tracking system, we may assume that the "best estimate" from the previous frame will place us within the valley, thus allowing AEFMA to converge to the global minimum. We also showed how the localization of the minima improves as the components of the σ-vector

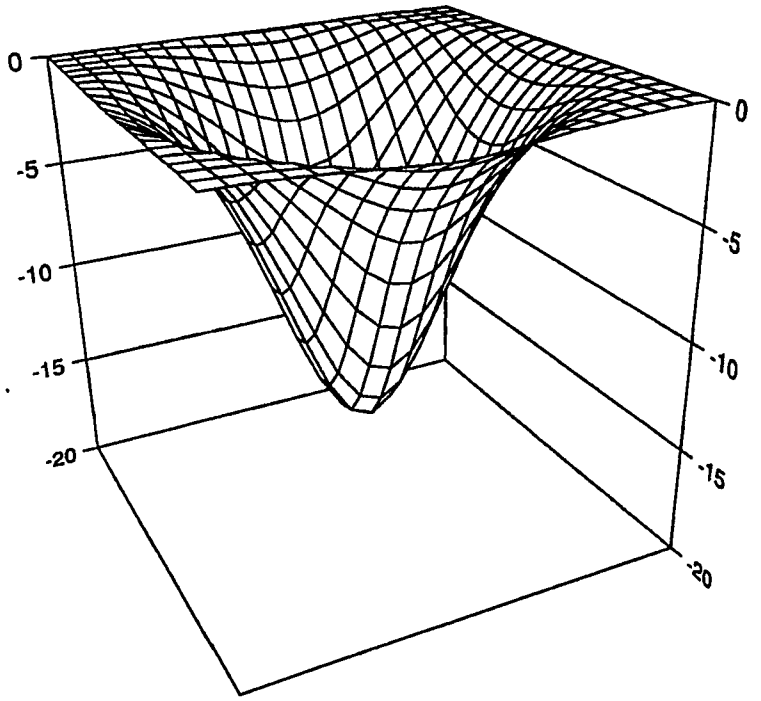


Figure 5.24. A 2-d slice of an actual CDF plotted versus the translational parameters x and y. The values of σ_x and σ_y are .1, on the large side.

are reduced. In the above examples, the components were reduced by a factor of 2 in the successive plots. In the figures it is easy to see that the minimum of the previous function easily falls within the valley of the succeeding plot. In fact, we have found this to be true even when the reduction factors go as high as 5. Thus, AEFMA's strategy of reducing the components of the σ-vector and then resolving for the viewing parameters is likely to succeed, especially when modest reduction factors like 2 are used. Thus, the terative reduction strategy employed by AEFMA permits accurate determination of the viewing parameters in spite of grossly erroneous initial guesses.

5.5.3 Optimization

The previous section showed that the CDF is a well behaved function that can be easily minimized by most continuous optimization methods. This section answers the

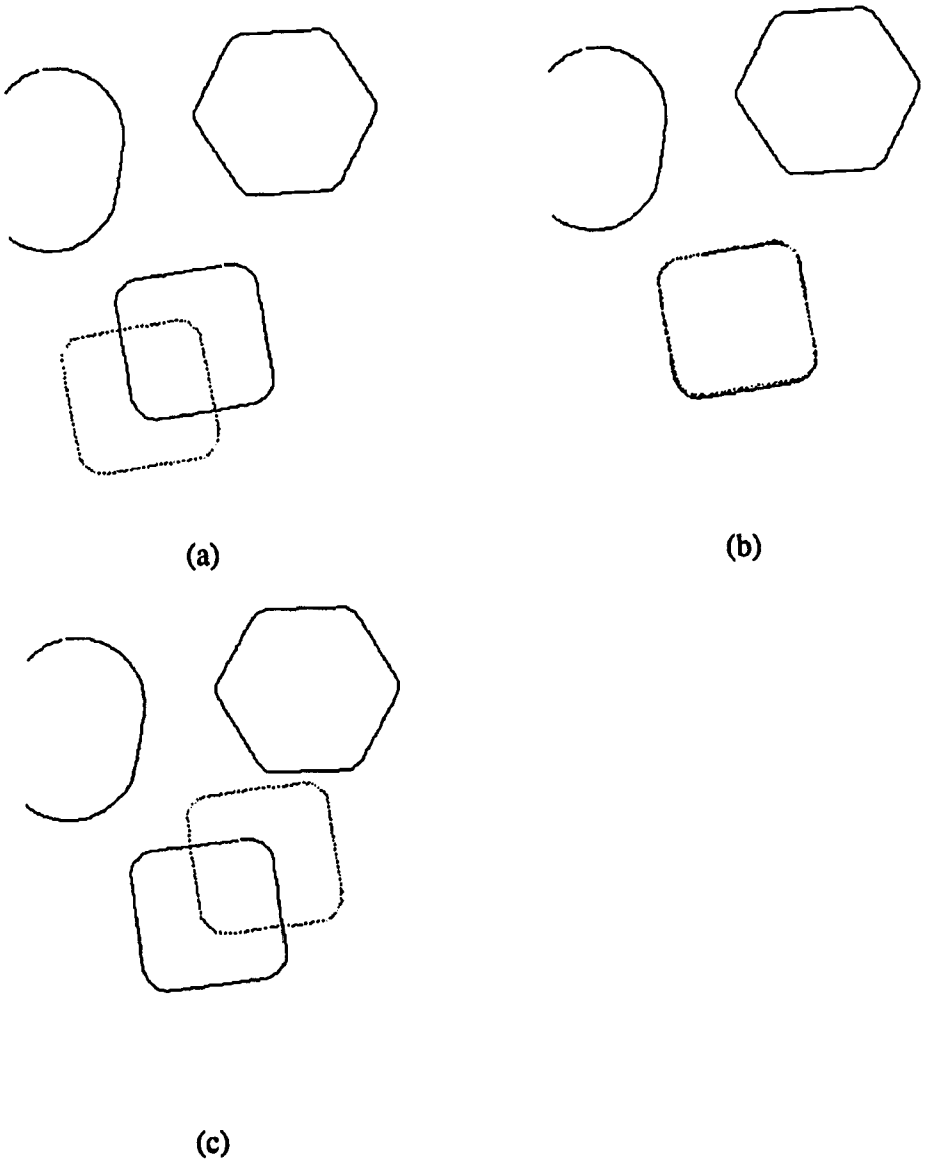


Figure 5.25. This figure shows the translational bounds used in Figs. 5.24, 5.26, and 5.27. Fig. (a) shows the lower extreme, i.e., $x = x_c - dx$, $y = y_c - dy$., Fig (b) shows the central value, i.e., $x = x_c$, $y = y_c$, and Fig (c) shows the upper extreme, i.e., $x = x_c + dx$, $y = y_c + dy$.

question of which continuous optimization method is best for our purpose.

In the early implementation of AEFMA the CDF was minimized using Powell's method [PFTV86]. This method uses no gradient information, only function evaluations.

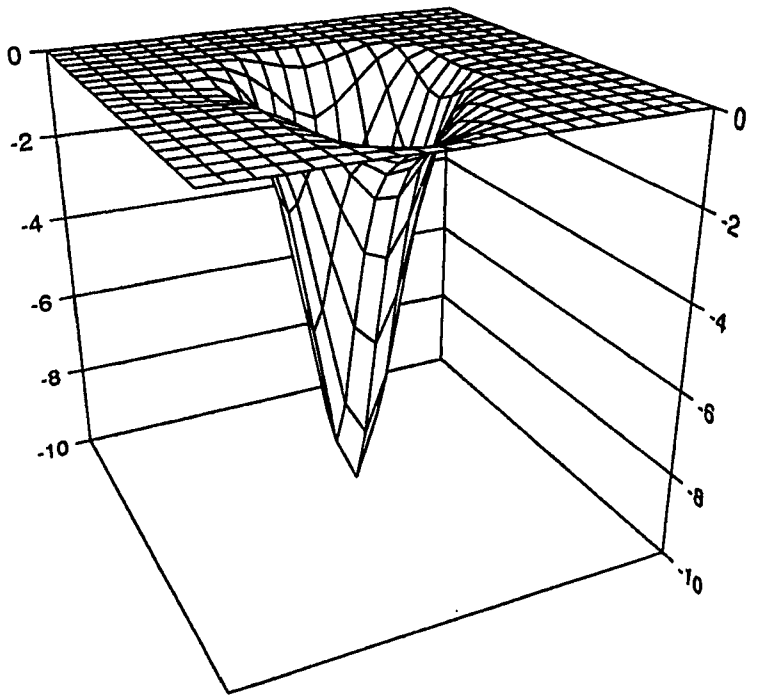


Figure 5.26. A 2-d slice of an actual CDF plotted versus the translational parameters x and y. The values of σ_x and σ_y are .05.

This initially seemed to be an advantage because the gradients would have to be calculated numerically.

Powell's method is very simple. A one dimensional minimization procedure is used to find the minimum along each of the coordinate directions. Then, after minimization has been performed along all coordinate directions, a test is done to see if the direction of the net movement over all of the coordinate minimizations is a promising direction along which to continue minimizing, as in the case of a long, thin valley in the objective function. If so, this direction replaces one of the coordinate directions. This process is continued with the new set of directions.

While Powell's method does work, our experience shows that the test to replace one of the current directions with a new direction is hard to satisfy and almost never occurs. Thus, in our case, Powell's method was reduced to a cyclic minimization along the coordinate directions. This leads to very slow convergence. To overcome this problem,

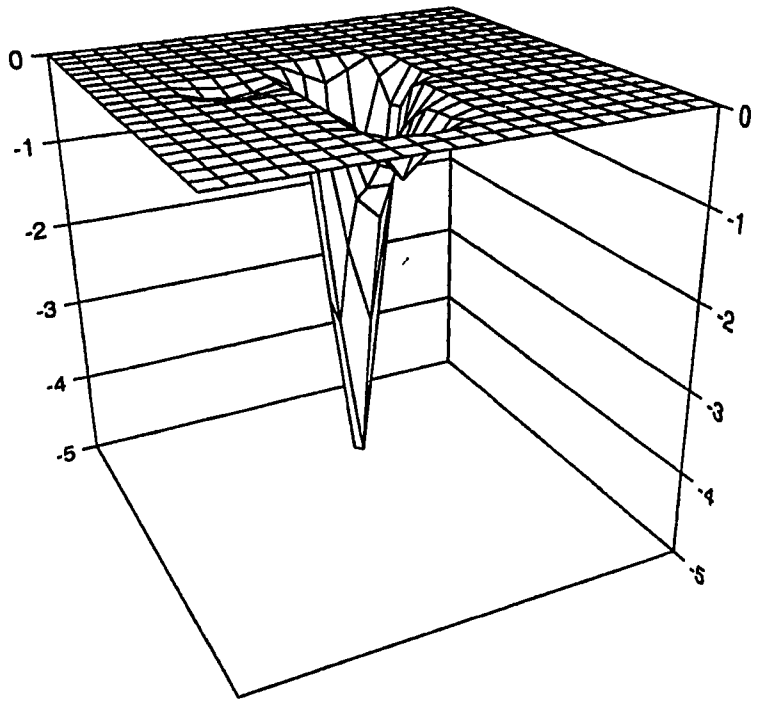


Figure 5.27. A 2-d slice of an actual CDF plotted versus the translational parameters x and y. The values of σ_x and σ_y are .025.

we tried the conjugate gradient method [PFTV86], which, as its name implies, requires the gradient of the function. However, in view of the often very slow convergence of Powell's method, the extra computation involved was expected to reduce the overall number of CDF evaluations.

The conjugate gradient method starts by evaluating the gradient of the function and performing a 1-d minimization along the direction of the gradient. The next iteration, the gradient is again computed, and is used in a simple formula, along with the previous direction, to compute the new "conjugate" direction along which to minimize. These conjugate directions give the conjugate-gradient method the property of *quadratic termination* where, if the objective function is a true quadratic, the conjugate gradient method will find the minima of the quadratic exactly, assuming infinite precision. For non-quadratic functions, this property implies quadratic convergence to the minimum, since all functions are approximately quadratic near a minimum. Unfortunately, with

finite precision, the conjugate directions can become linearly dependent as the algorithm progresses, thus improperly constraining the search for a solution. Thus, there have been improvements to the conjugate gradient method that prevent this from occurring. We used such a variant, due to Polack and Ribiere, which tends to reset the conjugate direction to the gradient, causing it to behave like steepest descent, when it is not near a minima or if the conjugate directions become singular. We have found that this approach works very well, often with fewer than 10% of the function evaluations of Powell's method.

The conjugate gradient method requires that the gradient of the CDF be supplied. Calculating the gradient of the CDF is not complicated. We used central finite differences, to compute the derivatives in each coordinate direction, i.e.,

$$\frac{\partial f}{\partial x_i} \approx \frac{f(x_i + dx_i) - f(x_i - dx_i)}{2dx_i},$$

as opposed to forward or backward differences, i.e.,

$$\frac{\partial f}{\partial x_i} \approx \frac{f(x_i + dx_i)}{dx_i},$$

because the central differences are more accurate, especially in the neighborhood of a minimum.

5.6 AEFMA's Relationship to Previous Work

In this section we examine the relationship AEFMA bears to previous work. We will first briefly reexamine the algorithms that use object-attached shape representations and contrast the way that they determine the viewing parameters of a model with the way that AEFMA determines the viewing parameters. We will then examine approaches that possess elements of similarity with AEFMA, and compare the relevant aspects of these algorithms and AEFMA.

5.6.1 Methods that use Object-Attached Shape Representations and AEFMA

Earlier in this chapter, we showed that AEFMA is able to estimate the viewing paramters of a model without using the object-attached feature assumption, thus allowing it to work with very generally shaped objects. Methods that rely on the object attached feature assumption, on the other hand, require that the objects possess certain types of 3-d features (see Table 1.1). The typical approach to viewing parameter estimation under the object-attached feature assumption is illustrated in Fig. 5.28. There, an imaging transformation with several unknown parameters, such as the one described by Eqs. 5.4 and 5.5, is applied to the model features, in this case the edges of a cube. Since the edges of a cube can be reliably expected to produce linear edge fragments when imaged, (invoking the object-attached feature assumption), such methods take advantage of this expectation to form a correspondence between a subset of the models 3-d features and the 2-d detected features. The correspondence is indicated by the lines connecting the cubes edges and the linear fragments in Fig. 5.28. The imaging transformation is usually modelled by perspective or, more commonly, weak perspective. Therefore, the imaging transformation can be described by a set of equations with six unknowns. The unknowns are determined by forming enough correspondences between 3-d model features and 2-d image features that the system of equations describing the imaging transformation is either completely determined or overdetermined. If the equations are completely determined, analytic solutions for the unknown parameters exist, [Hut88, LHF88, HLZ+87, FT86, FB81] for example. If the system is overdetermined, then either a consistent solution exists, or no consistent solution exists. Some algorithms, [Low87a] for example, assume regardless that the solution is consistent, and apply numerical methods to find the solutions of the resulting simultaneous nonlinear equations. Other methods, such as those based on the predict-observe-backproject paradigm, described in Section 3.2.3, simply add correspondences until the constraints associated with the new correspondences become inconsistent or uniquely determine the viewing parameters.

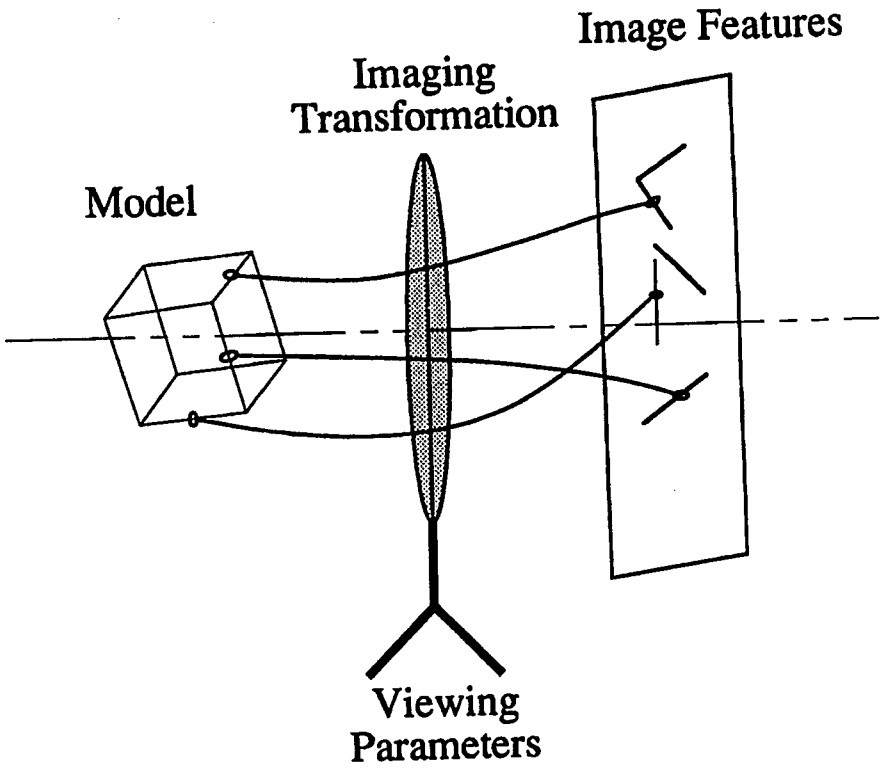


Figure 5.28. The typical model for viewing parameter estimation under the object-attached feature assumption. There, an viewing transformation is applied to the model to obtain the positions of the image features. The problem is to determine the viewing parameters that would cause the viewing transformation to map the 3-d model features onto the image features. In order to accomplish this, correspondences, indicated by the connecting lines between pairs of image and model features, are formed between 3-d model features and 2-d image features. The constraints imposed by the correspondences are then used in a number of ways, described in the text, to solve for the viewing transformation.

In the scenario of the previous paragraph, if the correspondences were erroneous, then the resulting estimates of the viewing parameters are meaningless. In this case, a new set of correspondences must be attempted, with the result that an exponentially large set of possible correspondences is being searched. Further, if the features being used violated the object-attached feature assumption, such as the example of Fig. 4.3, then the "constants" will change in the system of equations described above, i.e., the 3-d position and orientation of the 3-d features on the surface of the object will change, resulting in

meaningless solutions. Thus, such methods cannot be used to estimate the attitude of objects that do not produce object-attached features. Also, it does not appear that these methods can be extended to the general case, firmly grounded as they are in the use of 3-d to 2-d feature correspondences.

In contrast to the methods described above, AEFMA never makes explicit correspondences between parts of the 3-d model and of the 2-d image. Rather, AEFMA only compares the 2-d shape predicted by the model and the 2-d shape observed in the image. Further, AEFMA never makes hard correspondences between features in the predicted domain and in the observed domain. Instead, predicted primitives are allowed to match many of the image primitives to degrees determined by the value of the IDF. Thus, AEFMA can be thought of as forming "fuzzy" correspondences between predicted features (primitives) and image primitives. Forming fuzzy correspondences permits AEFMA to employ efficient continuous methods, rather than inefficient discrete methods, to solve for the best viewing parameters. In addition, since only 2-d predicted primitives, not fixed parts of the 3-d model, are being put into fuzzy correspondence with 2-d, AEFMA can be viewed as performing a simultaneous search of the space of viewing parameters *and* the surface of the 3-d model. This is a mode of operation that is fundamentally alien to approaches that rely on the object-attached feature assumption.

5.6.2 Related Methods

There are a number of vision algorithms with which AEFMA possesses some elements of similarity, or stimulated the author's thinking in the direction that led to AEFMA. In particular, the early attempts by Hemami *et al* [HW75] and Watson *et al* [WS82] to estimate viewing parameters by optimizing a function measuring the difference in the shape of predicted edges and the shape of image edges helped to inspire AEFMA.

Hemami *et al* used a simple disparity function:

$$E = \sum_{i=1}^{N} \left| \mathbf{x}_i^i - \mathbf{x}_i^p \right|^2 , \tag{5.6}$$

where $\mathbf{x}_i^i$, $i = 1 \ldots N$, are the locations of image edge points and $\mathbf{x}_i^p$ are the locations of the edge points predicted by the silhouette of the wireframe model. While Hemami's method does enforce correspondence between the image data points and the model data points (both are indexed by i in Eq. 5.6), it, unlike methods that rely on the object-attached feature assumption, forms the correspondence between the 2-d features in the predicted and the observed domains. Normally, this is difficult to accomplish since the correspondence would have to be recomputed each time a new value of the disparity function was computed. However, Hemami *et al* simply resampled the predicted silhouette boundary so that there were the same number of predicted and observed edge points, a simple enough operation to allow it to be done quickly. Unfortunately, for the algorithm to work properly this requires that no spurious edges can appear in the image, severely limiting the practical application of this approach. Hemami *et al* then attempted to find the minimum of Eq. 5.6 using the Gauss-Newton or the Newton-Raphson algorithm. They reported many problems with local minima. In spite of its shortcomings, this approach, more than any other, inspired AEFMA.

A similar method [WS82], which we have described in Section 3.2.5 of Chapter 3, and therefore will not describe in detail here, appeared after the work of Hemami *et al*. However, it is, in a sense, a step backward from Hemami *et al*. Watson *et al* used Fourier descriptors computed from the silhouette of the object as their basic shape representation, the shape disparity function was inherently global. Thus, they were able to completely sidestep the issue of correspondence, at the expense of an impractical approach. Hemami *et al* at least recognized that the issue must be dealt with if they expected to estimate the viewing parameters of objects with missing or distorted parts.

Recently, there have been a number of investigations into the problem of using *deformable* models as a means to recover properties of images. Typically, as we do in

AEFMA, these approaches attempt to minimize some type of pseudopotential. In the 2-d domain, Kass *et al* [KWT87] have used deformable contour models with an energy function that includes an internal "spline energy", essentially a smoothing term, and a term that describes "image forces" such as regions of high gradient, or contour endpoints, that the deformable contour should be attracted toward. Kass *et al* used a regularization-based approach to the minimization of the functional. This, and regularization approaches in general [PTK85, Bou87, LP88], proceed by applying the calculus of variations to the functional to obtain the Euler differential equation of the unknown function. In the case of Kass *et al*, the function was an arclength-parameterized contour spline. The differential equation is then discretized and solved by numerical methods.

Amini *et al* [Ami90, AWJed] have shown that there are many problems with the Kass *et al*'s regularization approach, including numerical instabilities and the inability to apply "hard" constraints, i.e., constraints that are active and affect the solution, or are inactive and have no effect on the solution (such constraints can be used to make the problem more stable). They show how to solve both of these problems by innovative use of dynamic programming. In particular, Amini *et al* apply the method to the case of 2-d contours, as in Kass *et al* above, and extend the approach to the recovery of 3-d surfaces from range images as well.

Terzopolous *et al* [TWK36, TWK87] describe an approach based on regularization for matching deformable models to intensity images. The models are continuous "tubes" that can blow up or shrink as necessary to minimize pseudopotential. The energy potential, as in the work of Kass *et al*, and Amini *et al* contains smoothing terms. In addition, Terzopolous *et al* include a term that tends to force the model into a symmetrical shape. The image influences the model by tending to attract occluding contours (points on the tube with tangents normal to the line of sight) to nearby regions of the image possessing large gradients.

The terms of the potential functions that are influenced by the image in the above

methods are computed in a very local manner, usually at the points that the current model suggests. Thus, there is no mechanism for image structures that may be distant from the current state of the solution to influence the solution. Thus, these methods may easily fall into local minima. Experiments were conducted with such a potential function in the course of this thesis. While such a potential function gave good results when the viewing parameters were known quite accurately to begin with, it invariably fell into local minima when started from more distant viewing parameters. We will compare this approach to AEFMA in Section 5.9. It should be noted that, since they are deformable, the models in the methods of Kass *et al*, Amini *et al* and Terzopolous *et al* are considerably less constrained, than the rigid models used by us. Such models have many degrees of freedom, making it more likely to provide a "way out" of a multidimensional valley. In the more constrained, 6-d parameter space in which AEFMA operates, problems with such potential functions forced us to develop the approach described earlier.

Solina *et al* [Sol87, SB90] have described a method for fitting deformable models of 3-d objects in the form of superquadrics [Bar81, Bar84] to range images. This method bears similarity to AEFMA because the superquadric models are more constrained than the deformable models described above, being described by 11 parameters. In contrast the deformable models of Kass *et al*, for example, has twice as many free parameters than it has points in the deformable contours boundary (twice since each point is 2-d). However, Solina *et al's* energy function is not very sophisticated, being essentially a point-by-point summation of the square of the distances of the range data points from the surface of the superquadric. Thus, this method is likely to be trapped by local minima if the initial estimates of the superquadric parameters are not close to an acceptable minimum. In spite of this potential problem, the results are reasonably good, given good initial estimates of the parameters. Further, the optimization method is a continuous method and converges quickly to correct results, lending credence to the argument for continuous versus discrete optimization.

5.7 Human Attitude Estimation

While not necessarily directly applicable to AEFMA, it is nevertheless interesting to consider what is known about how people perform the task AEFMA is designed to perform. For example, presented with an unfamiliar pose of an object, do people somehow key in on special features that they intuitively "know" are derived from a specific part of the object, and then use that knowledge to position the object in their minds, essentially instantaneously? In other words, do people use object-attached features? Or, do they mentally perform a search of the viewing parameters, comparing what they expect the object to look like to the pattern they have on their retinas? We believe that the second scenario is closer to the truth than the first, and there is clear psycho-physical evidence to support this claim.

Perhaps the most well known is the work of Cooper and Shepard [CS84]. They concluded that humans do use mental rotation of an object to attempt to match a mental model of a 3-d object with a 2-d retinal image. The manner they deduced this is convincing. They presented people with two images of 3-d objects and asked them to determine whether the drawings depicted the same object. The results showed that people displayed a response time that was linear in the angular difference between the two objects (when identical) to determine if they were the same. This implies that people are mentally rotating objects in order to compare them with the other. In fact, Cooper and Shepard inferred mental rate of rotation of 53° per second. This process of rotating a mental model in order to determine the position and orientation of maximum similarity between the mental expectation and the observed scene is very similar to the search of viewing parameter space performed by AEFMA.

Recently, Huttenlocher [Hut88] reported on interesting results contained in an unpublished paper by Tarr and Pinker [TP88]. They performed experiments with letter-like characters that possessed similar local features, ruling out recognition by feature indexing. An example of the stimuli used are shown in Fig. 5.29. Each character was given

Figure 5.29. Examples of the type of stimuli used in Tarr and Pinker's work [TP88].

a name and was learned in a reference position. Subjects were then asked to name characters presented at various orientations. The response time of the subjects was again linear in the angular difference between the presented stimuli and the learned reference. This supports the results of Cooper and Shepard, and provides more evidence that an AEFMA-like process is used by people when recognizing familiar objects in unfamiliar orientations.

Another interesting aspect of Tarr and Pinker's work is that as the subjects became familiar with the new orientations through practice, the response time became nearly identical with response for the reference, indicating that people "store" commonly encountered views of objects, thus reducing the amount of mental rotation necessary to recognize a stimulus. This process is very similar to Cyclops' approach of storing a multiview representation, using a fast associative database to generated initial hypotheses, which are then confirmed, in part, by using AEFMA to rotate the objects until the appearance predicted by the model matches, or fails to match, the appearance of the image. Thus it appears that not only is there pychophysical evidence that people use a process similar in function to AEFMA to compare and recognize objects, but that the overall Cyclops framework fits in with our knowledge of human recognition.

5.8 Models

We now turn to the problem of what are the best type of 3-d models to use in AEFMA, and how to predict the edge contours given such a model and a set of viewing parameters.

One approach would be to use a surface-patch, CAD style model. Unfortunately, this would be too slow unless it were implemented on a supercomputer (perhaps even then). The reason is that, in order to accurately predict edges from such models, they would first have to be rendered using realistic shading models, such as Goraud or Phong [Rog85] shading. Shading alone is a time consuming operation, especially if the model is complex, being comprised of many surface patches, and if the surface detail is rendered as well. However, we are only half done. Once the synthetic image has been generated we must apply an edge operator, perhaps the same one that is used to detect edges in the real image, to find the predicted edges contours. This also may be a slow process. Since the edge contours may need to be predicted from the model many times for each image, and given our goal of an implementation that runs on readily available hardware, this approach is impractical. While the continuing rapid decrease in the cost of a megaflop of computing power will make this approach practical within a few years, at the current time other alternatives must be sought.

AEFMA uses 3-d space curves to model objects. In contrast to surface patch models, predicting edges with space curves is very simple. All that is necessary is to project each 3-d point of the space curves using Eq. 5.5. Since connectivity is preserved in the projection process, the result is a non-uniformly sampled set of 2-d curves. These curves form the predictions of the edge contours in AEFMA. The contours are resampled using exactly the technique described in Section 4.3.2.3, resulting in edge contours that are represented in the dual (x, y, θ)-s representation also described Section 4.3.2.3. In order to speed the calculation of the tangent lines used to compute the attributes of the shape primitives, the space curves are augmented with the 3-d directions of the space curve, permitting the slope angles of the predicted edges to be computed very quickly.

While space curve models have the advantage that edge contours may be predicted very quickly, they have the drawback that they do not always accurately predict the appearance of the edge contours, especially where the object is self-occluding. Being of infinitesimal thickness, a simple space curve cannot model the effects of self-occlusion. A simple way of overcoming these difficulties would be to simply have a space curve representation for a number of possible views. We did not implement this in AEFMA, as we have encountered what we consider better approaches in the literature. Below we briefly describe the two most promising such approaches.

The problem of *quickly* and *accurately* predicting the appearance of edge contours from any viewpoint is a topic of active research that is largely beyond the scope of this thesis. However, there has been promising work lately on this problem. In particular, there are two approaches that we believe are applicable. The first is due to Van Hove [Van87b], while the second is due to Basri and Ullman [BU88].

Van Hove's approach has the advantage that it allows fast computation of silhouettes of objects, and with some extensions contemplated by the author, should be able to allow the prediction of internal edges that are the result of depth discontinuities. The approach is based on the parameterization of the surface points of an object in terms of the normal directions at the points. In this regard, Van Hove's representation is closely related to extended Gaussian images [Hor83]. Silhouettes can be defined as the points on the object that possess normals that are perpendicular to the line of sight. If the possible normal directions are represented by the surface of the unit sphere, as shown in Fig. 5.30, then, under weak perspective, the points that comprise the silhouette have normals that fall on the great circle whose plane is perpendicular to the line of sight. The model's silhouette generator is easily found by plugging in the normal values into the Van Hove representation, which returns the actual 3-d points. Then, the silhouette generator is projected to obtain the 2-d silhouette that we can use as a prediction of the edge contours, as shown in Fig. 5.30.

(a) (b) (c)

Figure 5.30. Given a model of an object, (a), the 3-d space curve, (b), i.e., the silhouette generator is often extremely complex. If the points of the surface are parameterized by the latitude and longitude angles, α and β, of the surface normal at the point, the silhouette generator corresponds in the Van Hove representation simply to points on a great circle, (c), of the α-β sphere.

While Van Hove's approach allows the fast prediction of silhouette edge contours, it is not able to predict edges that arise from other mechanisms. The method of Basri and Ullman is superior in this regard. Basri and Ullman's representation is an extension of AEFMA's space curve models. The method is based on approximating certain neighborhoods of the surface of the object by the circle of curvature, as shown in Fig. 5.31. The surface that is approximated is the surface that is near the edge contour generator when the object is seen from a particular view. Simple as this approximation may be, it nevertheless allows the appearance of edge contours to be predicted accurately over a wide range of viewing angles. However, the approximation is not valid over *all* viewing angles. Thus, several such models are required to obtain adequate predictions over all viewing angles. However, the number of models needed is considerably fewer than would be needed if simple space curves alone were to be used, since Basri and Ullman's models are valid over a much wider range of viewing angles.

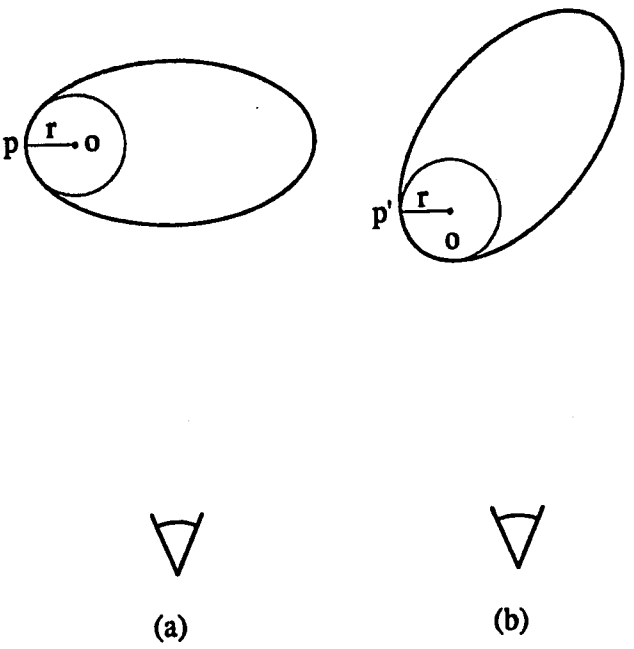


Figure 5.31. In (a) the point p is viewed on the silhouette of the object. The surface of the object in the neighborhood of p is approximated by the circle of curvature passing through p, with center O. The vector from the center of the circle to p is $\mathbf{r}$. In (b), the viewpoint has changed, so that the new silhouette point is p'. However, $\mathbf{r}$ is the same as before. Thus, p' can be computed from p from the equation $\mathbf{p'} = \mathbf{R}(\mathbf{p} = \mathbf{r}) + \mathbf{r}$, where $\mathbf{R}$ is the rotation matrix describing the rotation of the object.

Basri and Ullman's models are created by taking five images: a central image and four auxiliary images, and processing the edges appearing in them to create the models. The auxiliary images are taken by rotating the object equal rotational increments about the x and y axes on either side of the central image. From this information, the 3-d position, and magnitude of the curvature vector of the space curves comprising the model can be computed. After the model has been created, it can be used to accurately predict the appearance of edge contours over a wide range of viewpoints with little more effort than that necessary using simple space curves. We implemented Basri and Ullman's approach and applied it to create a model of NASA's space shuttle, including some of the surface

detail[2]. Fig. 5.32 shows the result. The model has been rotated 15° about the x axis, and superimposed on the actual edge contours derived from images of the shuttle after it had been rotated $\pm 15°$ about the x axis. As can be seen, the accuracy of the prediction is quite good. Further, much of the shuttle's internal detail is represented. This is an advantage of Basri and Ullman's method over Van Hove's. For this reason we feel that Basri and Ullman's approach holds the most promise as a practical method for fast edge contour prediction.

5.9 Experiments in Attitude Estimation and Tracking with AEFMA

AEFMA has been implemented in the C programming language on an Apollo workstation. This section describes the results of using AEFMA to estimate viewing parameters of objects in a number of images. Whether for use as part of a recognition system, or as part of a tracking system, AEFMA's ability to correctly estimate the viewing parameters of models in spite of poor initial estimates is critical to robust performance of any system in which AEFMA is included. Thus, we demonstrate AEFMA's wide range of convergence via the experiments. As we have mentioned earlier, as part of the Cyclops framework, AEFMA's wide range of convergence permits us to reduce the number of views stored in a multiview model (see Chapter 4). In a tracking scenario, AEFMA's convergence permits an object that has been lost, as it travels behind some obstacle for example, to be readily reacquired.

5.9.1 Methods

We acquired a number of 416×8 images of a plastic model of the space shuttle, such as the one in Fig 5.33, from various poses, with "cosmic" occlusions being mimicked

[2] This work was done at KMS Fusion, Inc.

by various laboratory objects such as lenses and construction paper. We used the shuttle to test AEFMA because it is largely smooth, and would likely present conventional algorithms with difficult problems if asked to estimate its viewing parameters. Images of smooth geometric shapes were also acquired, such as the ones in Fig. 5.34. These were used to characterize AEFMA's behavior in the presence of other objects, and differently shaped objects.

The images are uniformly preprocessed with our implementation of Canny's edge detector [Can83]. Using our array processor implementation, the edge detection takes about two seconds. Following this, we apply a simple linker, described in Section 4.3.2.2. The resulting edge contours are uniformly resampled and placed into the dual (x, y, θ)-s representation, described in Section 4.3.2.3. Then, we compute the shape primitives and the overall shape representation, described in Section 5.3. The shape primitives are then used to create the k-d tree-based detected_shape_database appearing in the pseudocode in Sections-pseudocode, which, recall, is used to quickly extract the image primitives that will result in a non-negligible IDF when paired with a predicted primitive. At this point, a model was put in some initial pose, and the algorithm run.

As described earlier, we are using space curve models to predict the appearance of edge contours under a particular viewing transformation. As discussed in Section 5.8, space curves were the only available option that would permit the algorithm to run at an acceptable speed. However, as also discussed in Section 5.8, a number of promising improvements to the space curve model are on the horizon, which would allow fast and accurate prediction of edge contours without requiring a supercomputer.

The models were aquired by taking an image of a model from a canonical viewpoint, usually directly overhead, and forming a 3-d contour that has the same x-y components as the edge contour in image coordinates, and a z component of zero.

5.9.2 Results

In the following sequences of images, the first image is the postscript-printer dithered rendition of the grayscale image. The remaining images, show the edges of of the image as solid black curves, and the model's predicted edge contours as dotted curves. The images progress in a sequence from upper left to lower right. The first non-graylevel image in a sequence shows the initial pose of the model. Following it are one or more intermediate poses generated in the process of the run. The last image in a sequence shows the contours predicted from the final pose of the object. The captions of the figures provide information and comments specific to that sequence. Absolute pose paramters will rarely be given. What is more informative is the difference between the initial and final poses. These numbers will be given in the form as a *pose delta vector*, or PDV: $(\alpha_f - \alpha_i, \beta_f - \beta_i, \theta_f - \theta_i, s_f/s_i, x_f - x_i, y_f - y_i)$.

The initial group of sequences tests AEFMA's range of convergence to an unobscured model of the shuttle. This was done by performing six runs where the pose of the model was varied over a wide range. The vector of viewing parameters of the initial pose, $(\alpha, \beta, \theta, s, x, y)$ differed from the final pose by as much as 50° in α, 180° in β and θ, a factor of 3 in s, and more than .1 in x and y.

The next experiment, shown in Figs. 5.41 to 5.45, demonstrates how AEFMA can be used for tracking. In addition, this group shows that AEFMA works successfully when the object is partially occluded. The scenario is that the shuttle is in space and is moving and rotating when it passes behind a piece of the space station (construction paper). The shuttle was placed in the initial image at a pose that approximates the pose difference between the shuttle in the succeeding images. After AEFMA determines the viewing parameters of the shuttle in the first image, these final viewing parameters are used as the *initial* guess in the following frame, and so on, for all five frames. Of course, using the last frame's result is not the best approach to tracking since it does not take into account the object's dynamics, which would allow a much better estimate to be made.

However, since AEFMA will only do better with a better initial estimate, this provides a lower bound on what is possible.

In this experiment, the object was moved while it was resting on a plane parallel to the cameral. Thus, save for the intial frame, α of the PDV tends to be small while s tends to be near 1.0.

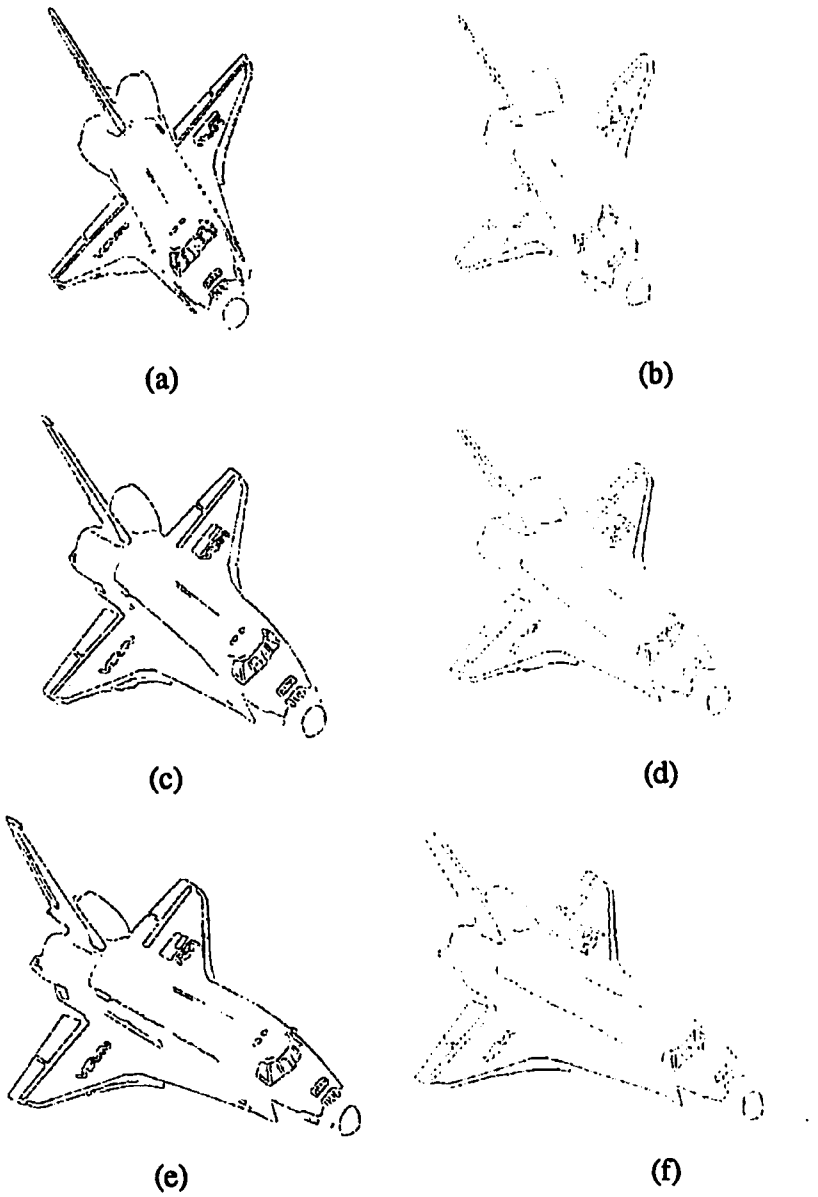


Figure 5.32. Above is the result of creating a model of the space shuttle using Basri and Ullman's method. The edge contours detected in actual images of the space shuttle are shown in (a), (c), and (e). The shuttle was rotated in increments of $\pm 15°$ about the y axis. On the right, in (b), (d), and (f) are the edge contour predicted by the model for the same viewpoints. As can be seen, the model does a good job of predicting the appearance of edge contours, both those comprising the silhouette boundary and those comprising internal details.

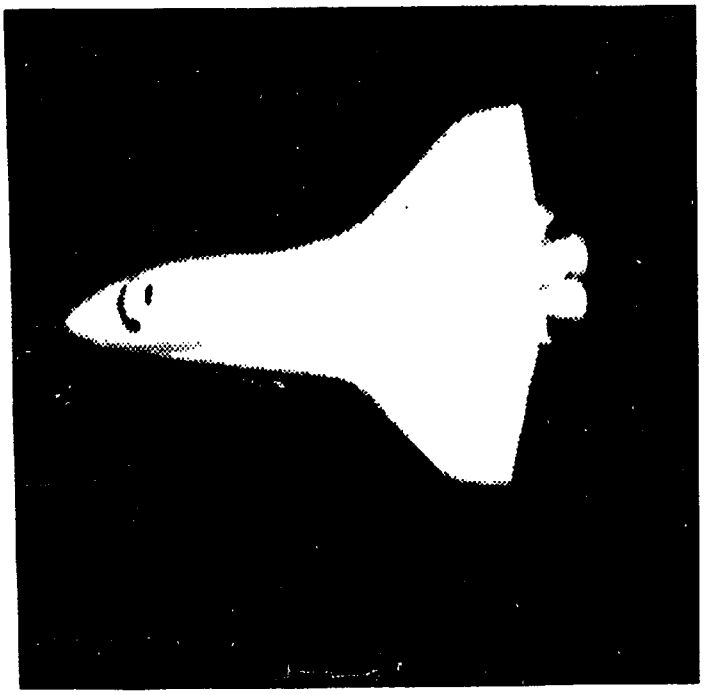


Figure 5.33. An image of the space shuttle.

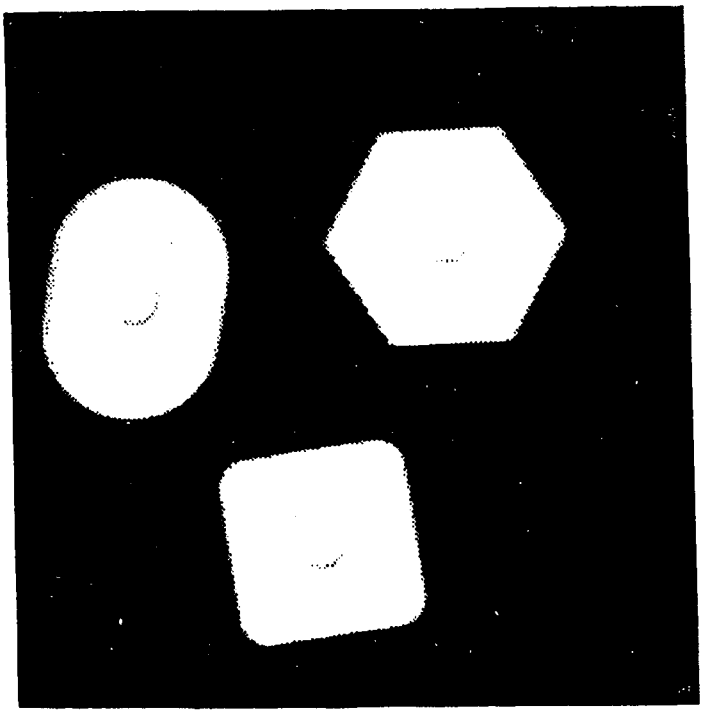


Figure 5.34. A few of the smooth geometric parts used in other experiments with AEFMA.

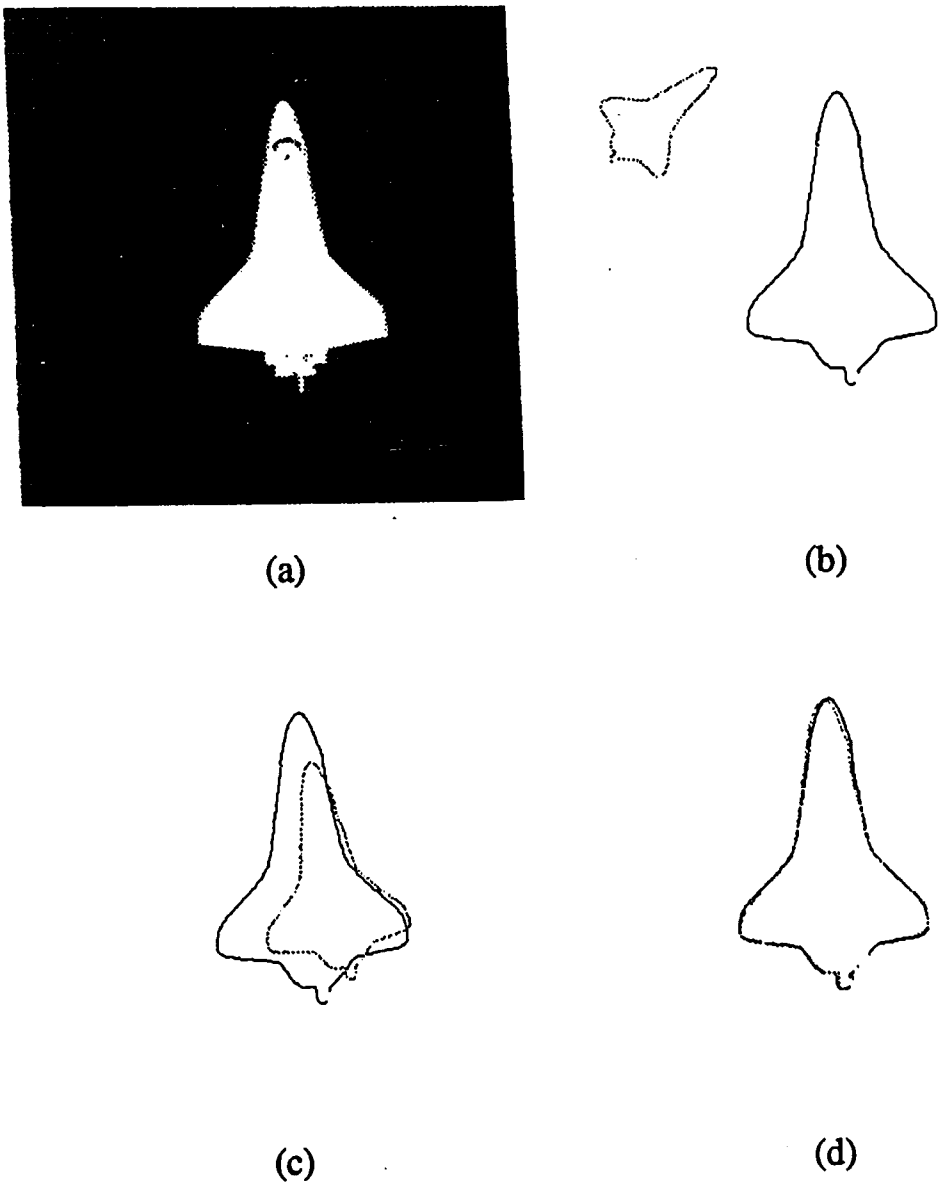
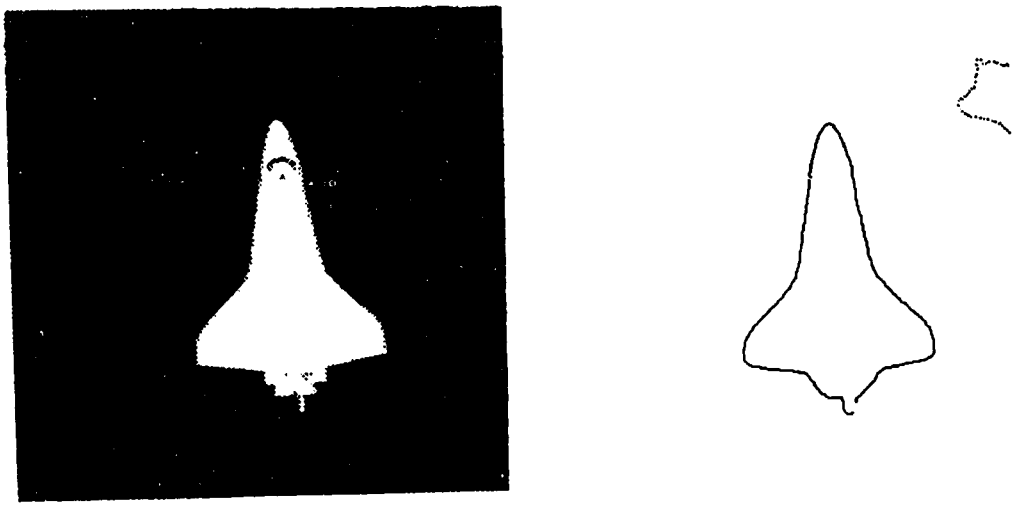


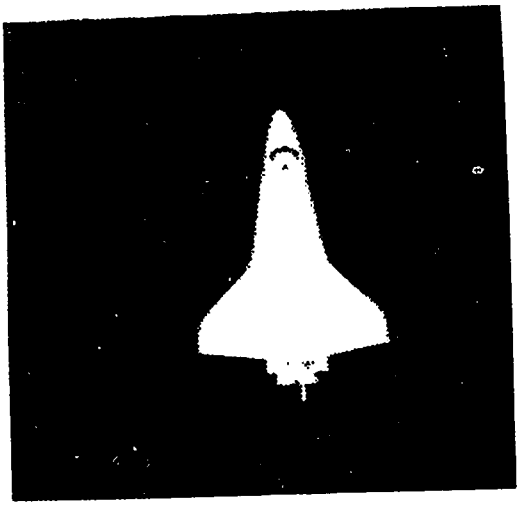
Figure 5.35. Run one testing the convergence of AEFMA. The PDV for this run is $(0°, 0°, 53°, 2.1, -.65, .44)$.



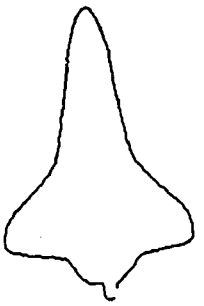
(a) (b)

(c) (d)

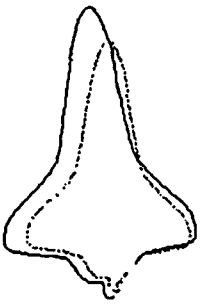
Figure 5.36. Run two testing the convergence of AEFMA. The PDV for this run is $(0°, 50°, 100°, 2.2, -.6, .53)$.



(a)



(b)



(c)

(d)

Figure 5.37. Run three testing the convergence of AEFMA. In this case, the initial pose of the model causes its predicted contour to be off the upper right of the image. The PDV for this run is $(20°, 30°, 31°, 3.4, .58, -1.12)$.

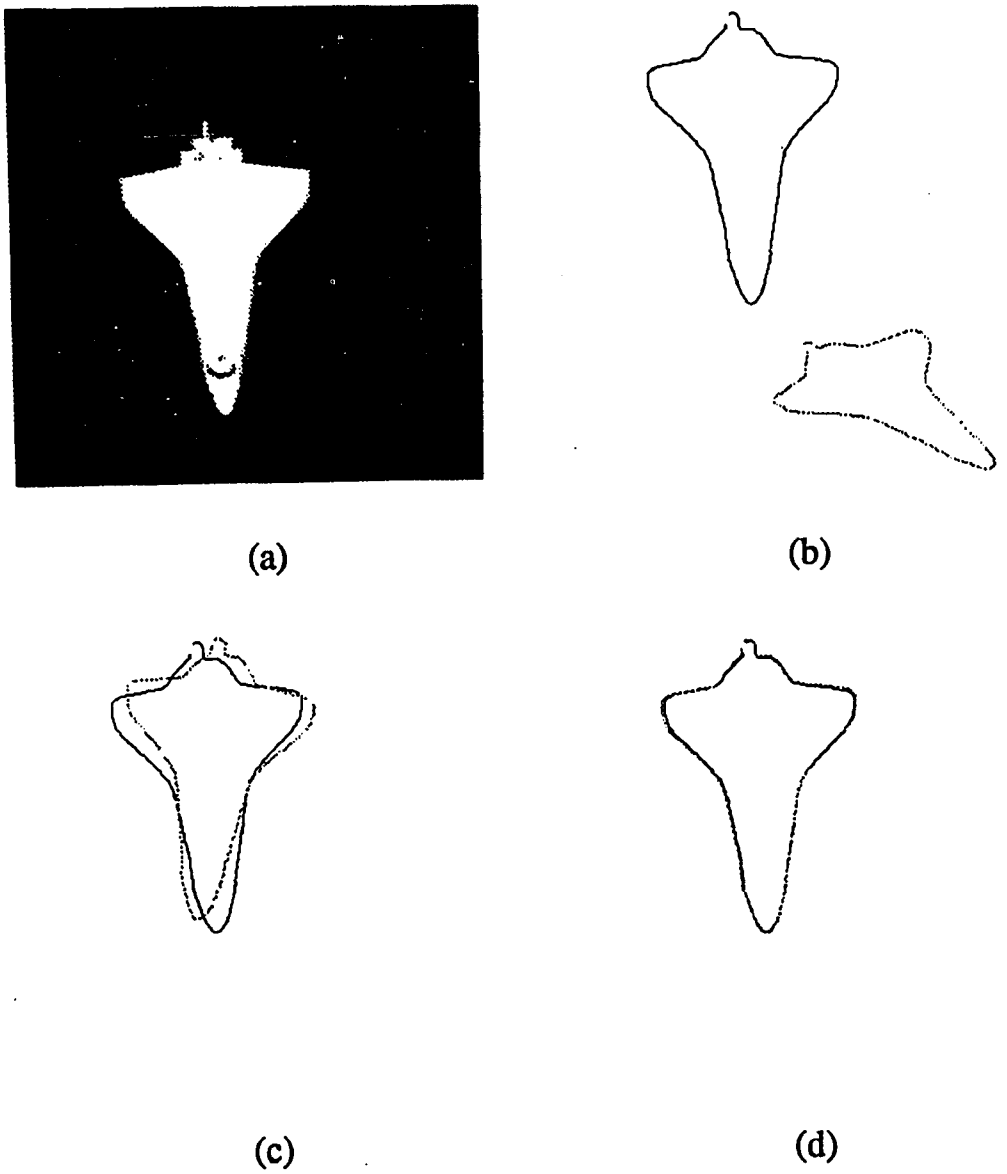


Figure 5.38. Run four testing the convergence of AEFMA. The PDV for this run is $(54°, 38°, 2°, 1.05, .45, .88)$.

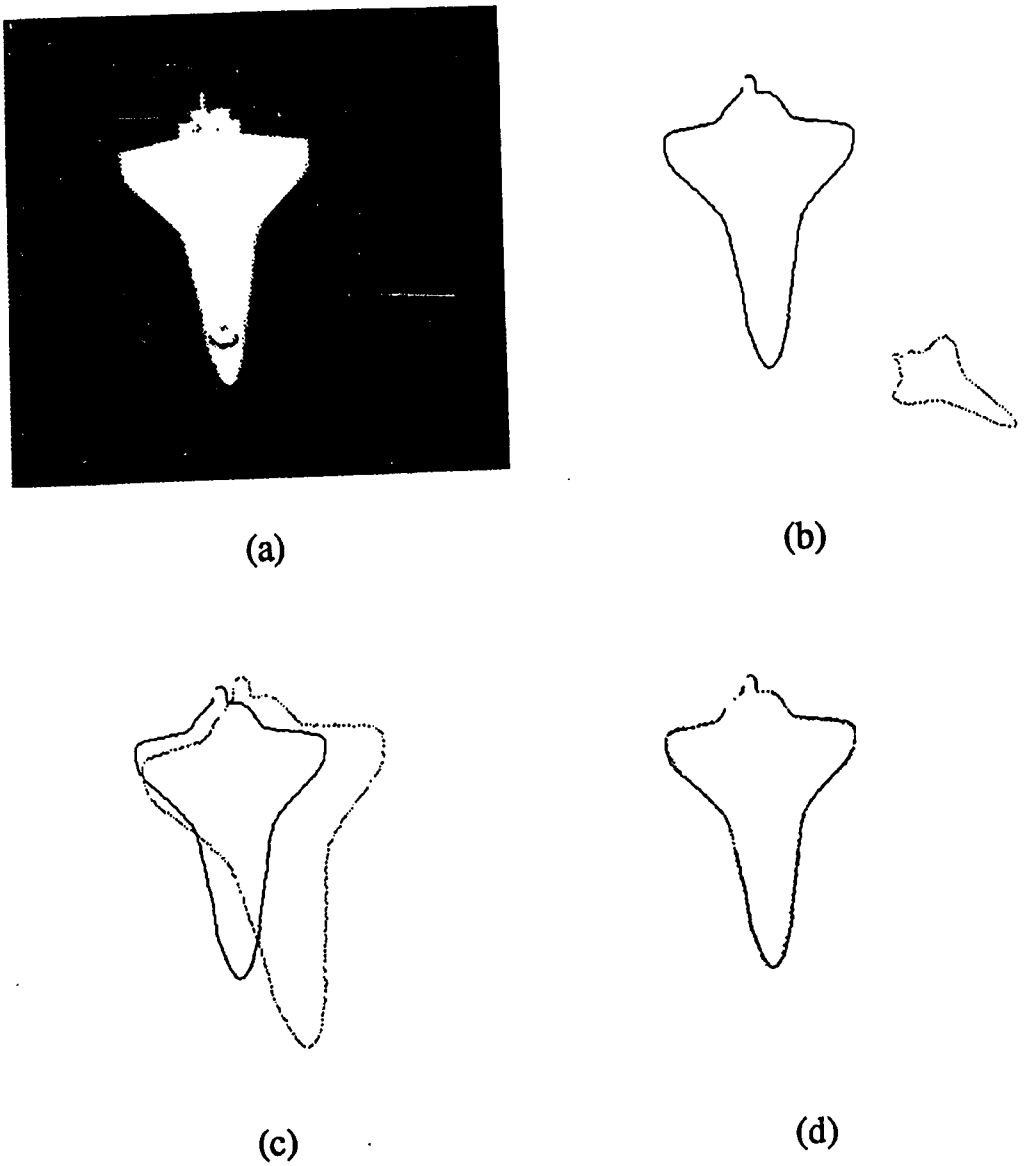


Figure 5.39. Run five testing the convergence of AEFMA.

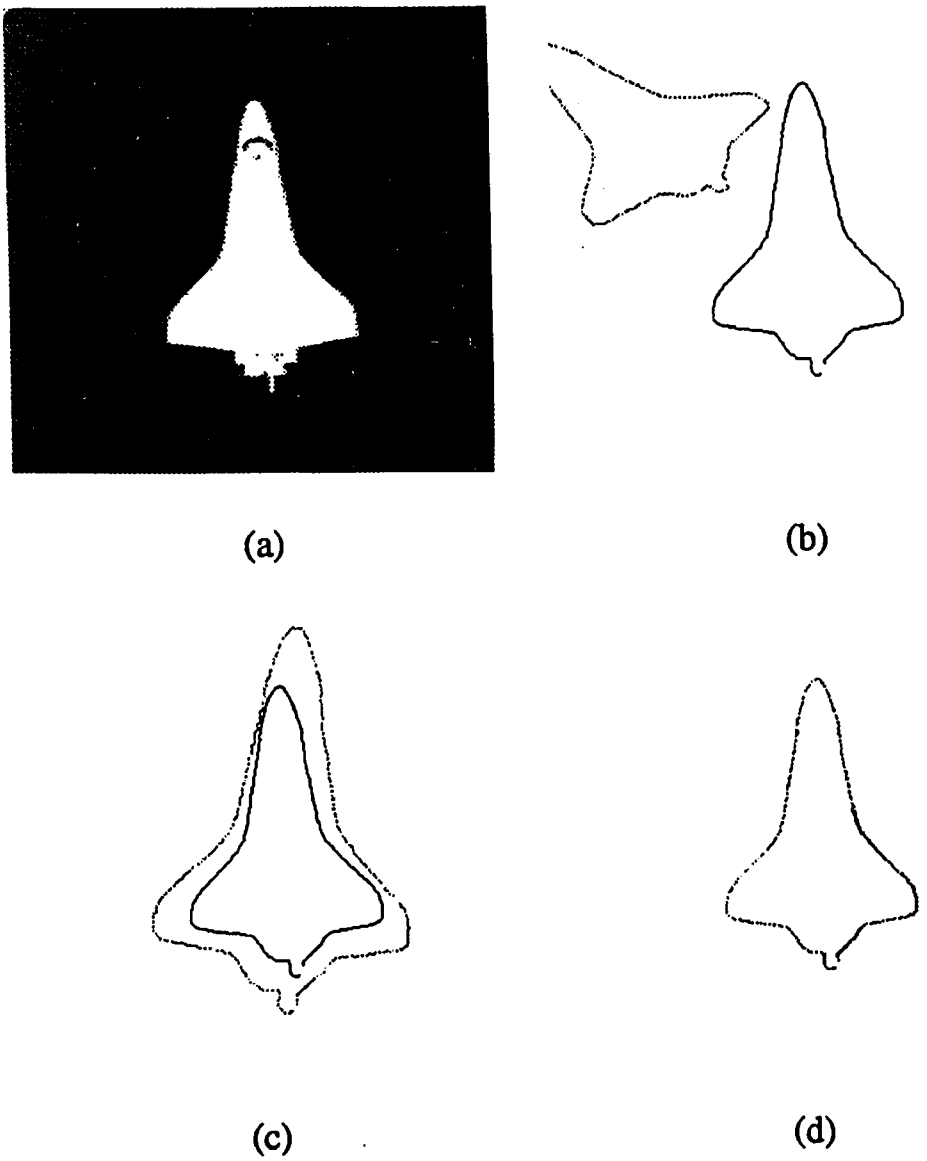


Figure 5.40. Run six testing the convergence of AEFMA. The PDV for this run is $(50°, 16.5°, 17.3°, .83, .65, -.89)$.

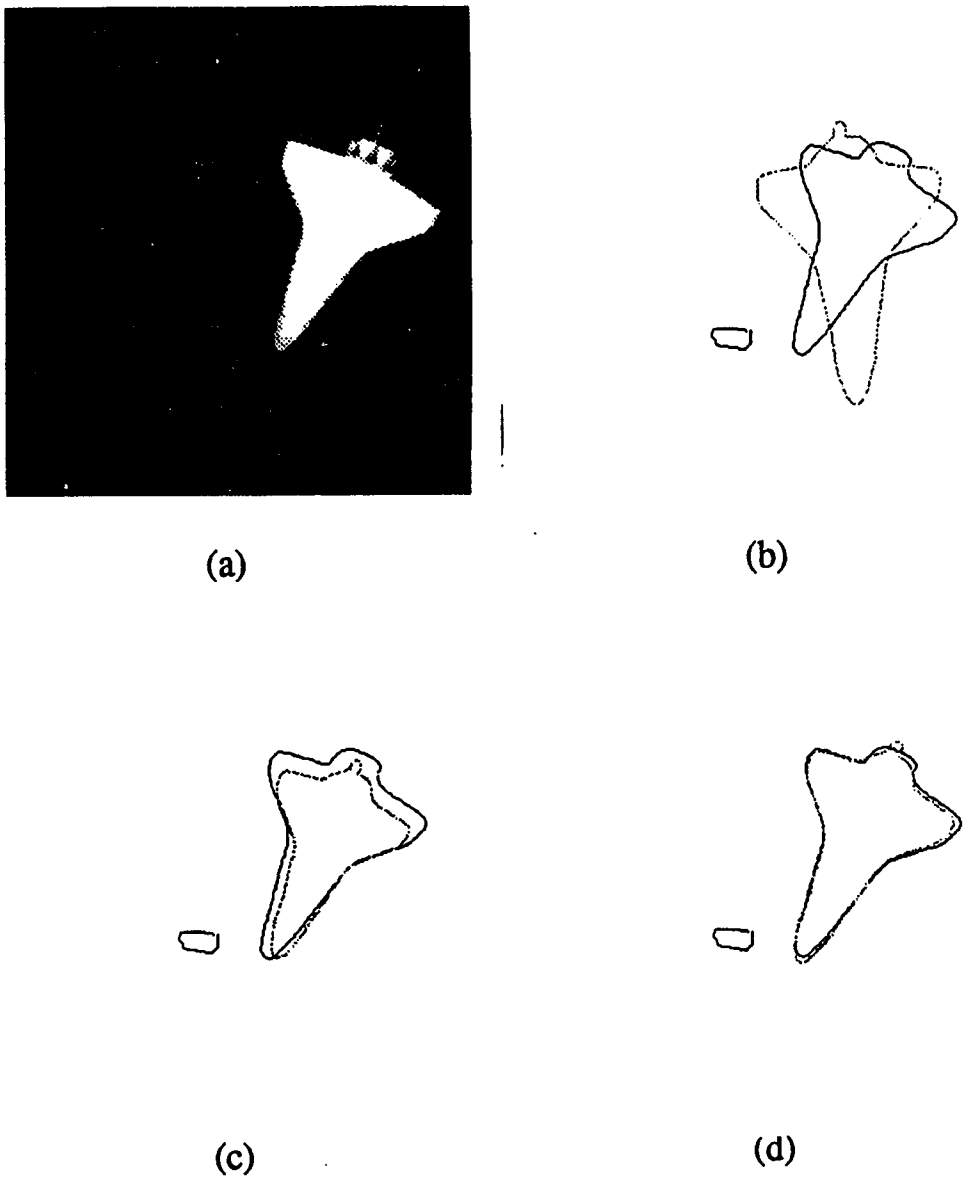
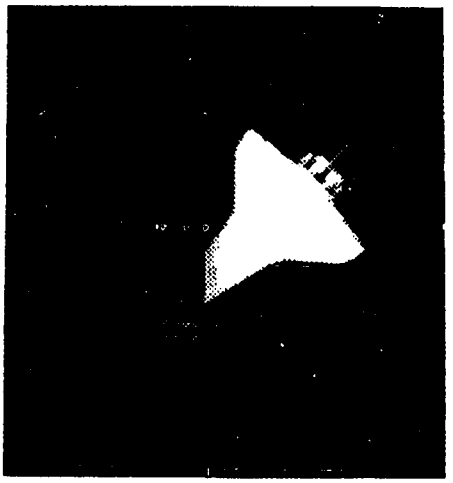
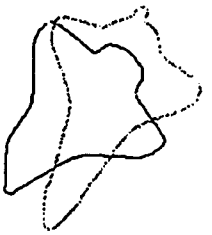


Figure 5.41. Initial frame in a set of five. The PDV for this run is $(10°, -14°, -13°, .86, -.01, .05)$.



(a)

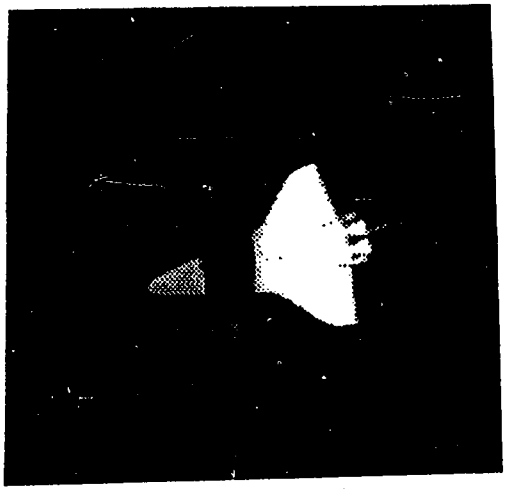


(b)

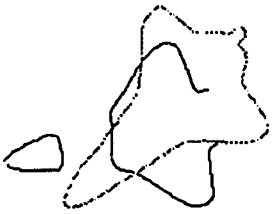
(c)

(d)

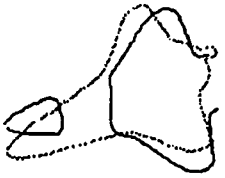
Figure 5.42. Second frame in a set of five. Note that the first initial pose of the model, shown in (b), is the same as the final pose in frame one, i.e., the pose shown in Fig. 5.41. Also note that the shuttle has become partially occluded in this frame. Nevertheless, AEFMA is able to track it. The PDV for this run is $(5°, 11°, 10°, 1.04, -.17, . - 05)$.



(a)



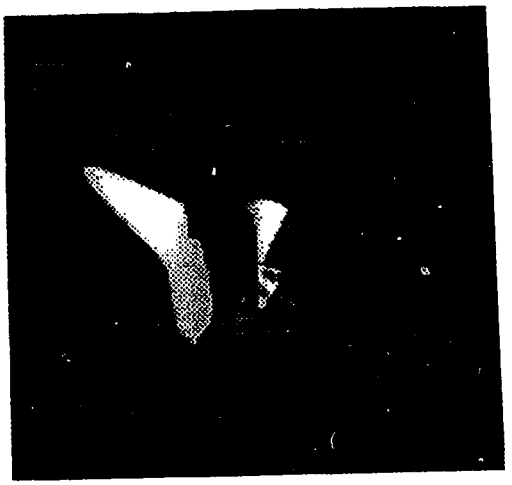
(b)



(c)

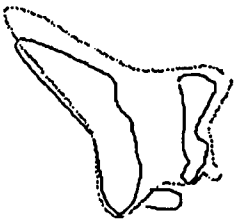
(d)

Figure 5.43. Third frame in a set of five. The initial pose of the model in this figure, shown in (b), is the same as the final pose in frame one, i.e., the pose shown in Fig. 5.42 (d). The severity of the occlusion increases in this frame. The PDV for this run is $(-2°, 14°, 15°, .98, -.14, . - .07)$.

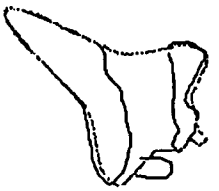


(a)

(b)

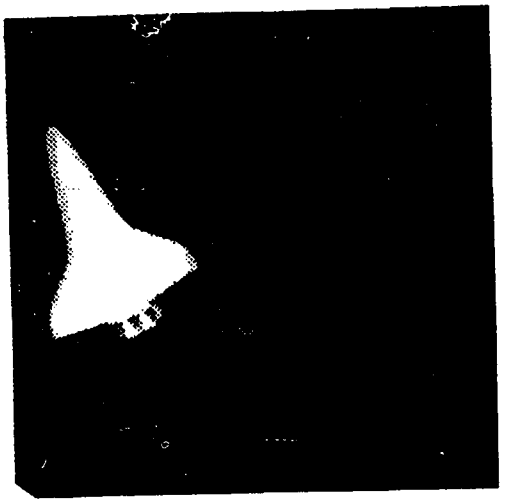


(c)

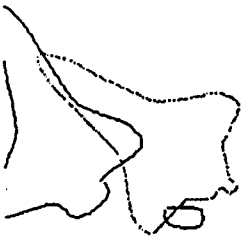


(d)

Figure 5.44. Fourth frame in a set of five. The initial pose of the model in this figure, shown in (b), is the same as the final pose in frame two, i.e., the pose shown in Fig. 5.42 (d). The PDV for this run is $(1°, 22°, 27°, .98, -.28, .-08)$.



(a)



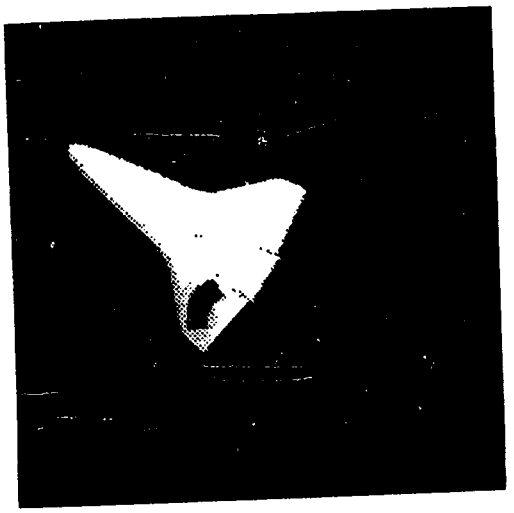
(b)

(c)

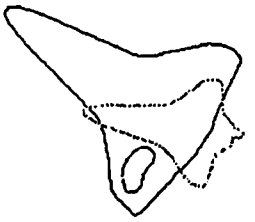
(d)

Figure 5.45. Fifth frame in a set of five. The initial pose of the model in this figure, shown in (b), is the same as the final pose in frame four, i.e., the pose shown in Fig. 5.44 (d). The PDV for this run is $(1°, 16°, 14°, .98, -.32, .05)$.

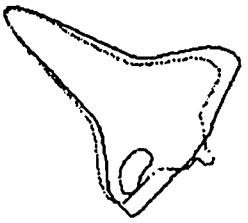
The next group of sequences, shown in Figs. 5.46 to 5.49, is analogous to the previous tracking experiment. This experiment simulates a docking manuver in space. The problem faced in docking is that as the docking site is approached more detail appears, while large structures leave the field of view. In this experiment, we simulate this by putting a sector-shaped mark on one of the wings of the shuttle. The mark is the "docking site". Initially, the shuttle is far away and we obtain the best estimates of the object's viewing parameters by using a model of the larger object in the scene, i.e., the shuttle. In the subsequent frames, the shuttle approaches more closely, and much of its outline leaves the field of view. At this point it becomes appropriate to use a model of the docking site, i.e., the mark, to determine the viewing parameters. We assume that the relative pose of the docking site (the mark) and larger structure (the shuttle) are known *a priori* so that the viewing parameters can be expressed relative to either coordinate system. This experiment also shows how AEFMA might be used with heirarchically organized models.



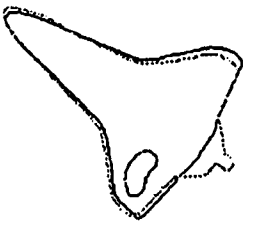
(a)



(b)



(c)



(d)

Figure 5.46. Initial frame in a set of four in a docking scenario. The "docking site" is the mark on the shuttles wing. Since the shuttle is relatively distant, far more accurate estimates of the viewing parameters result if the shuttle model is used rather than a model of the docking site. The PDV for this run is $(-5°, -13°, 13°, 1.72, -.17, .16)$.

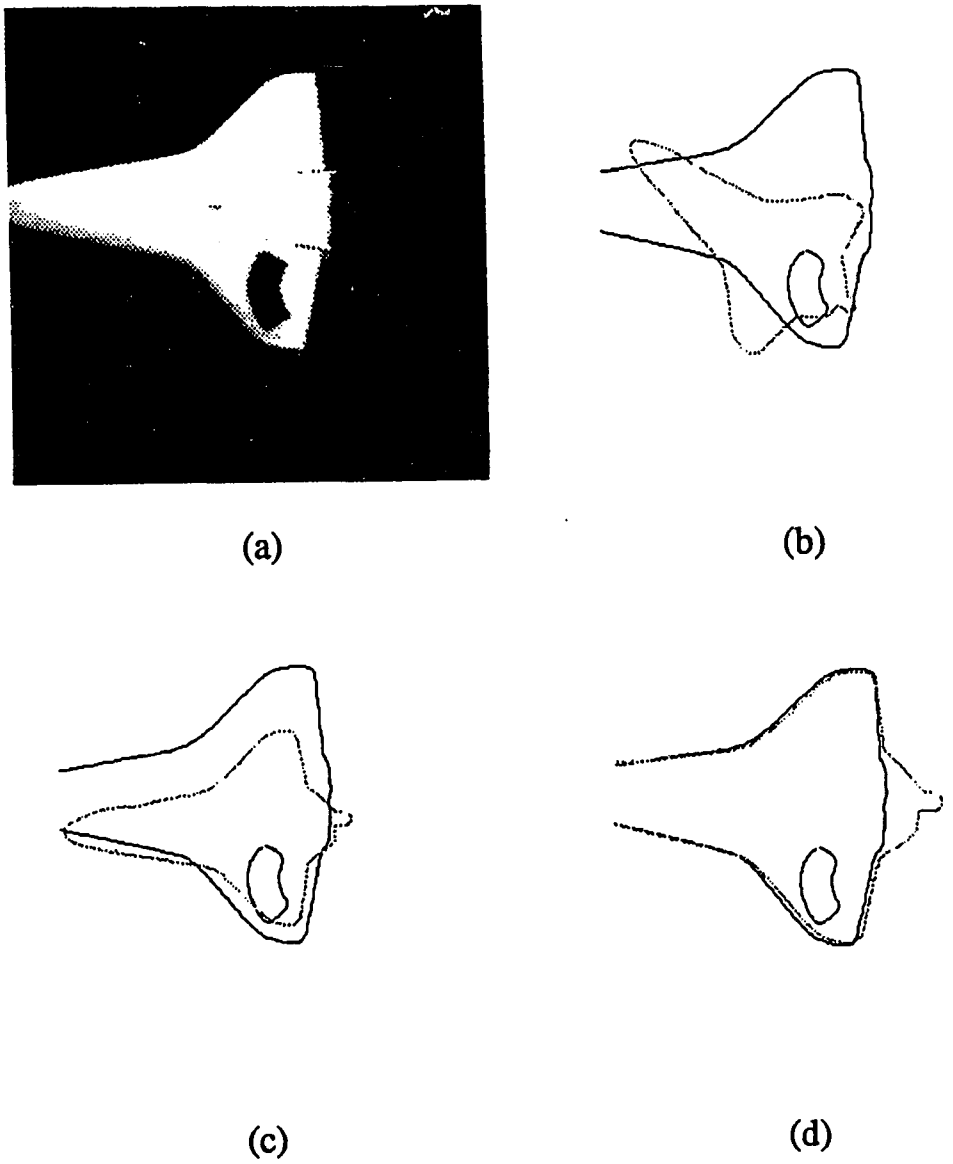


Figure 5.47. Second frame in a set of four in a docking scenario. Since the shuttle is still relatively distant, we continue to use the shuttle model. In a manner similar to the tracking experiment, the initial pose of the model in this figure, shown in (b), is the same as the final pose in frame four, i.e., the pose shown in Fig. 5.46 (d). The PDV for this run is $(6°, -17°, 17°, 1.44, -.09, .09)$.

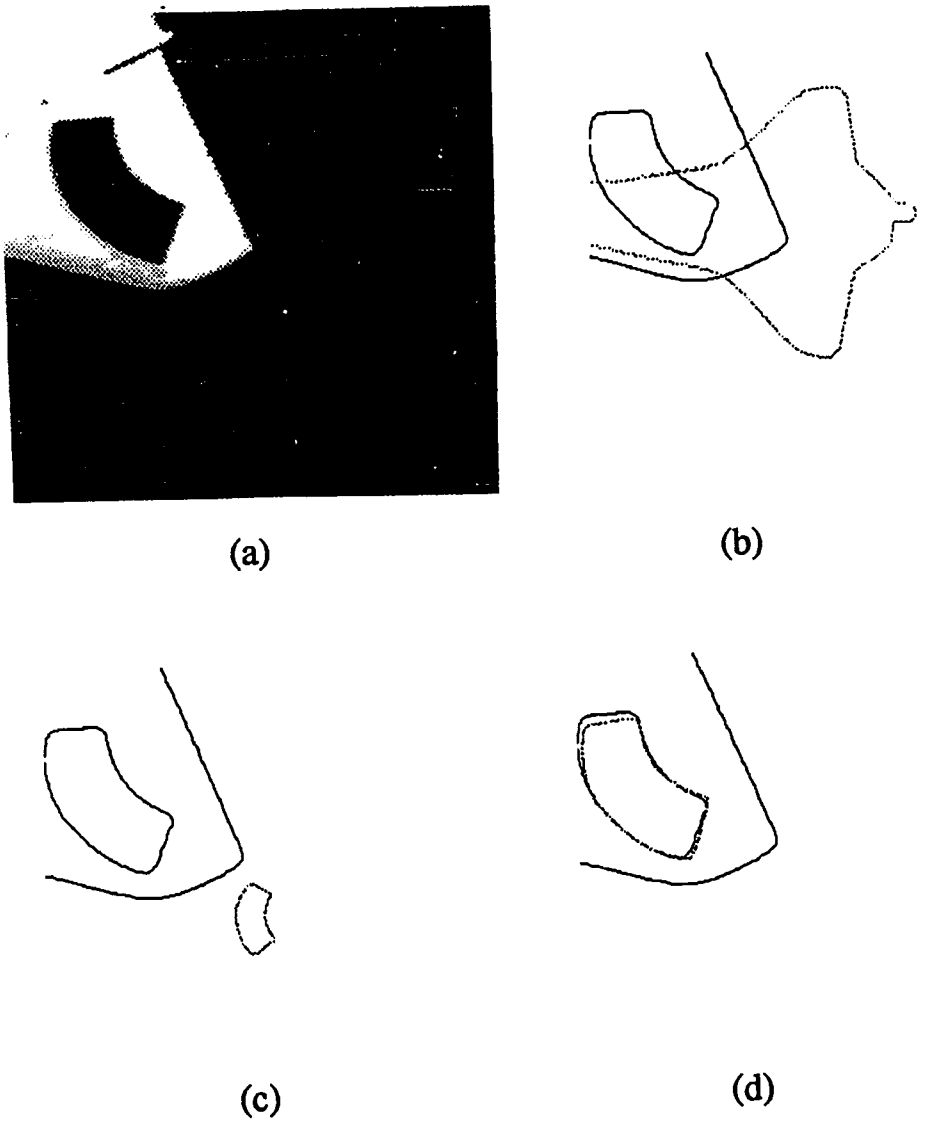


Figure 5.48. Third frame of four in a docking scenario. In this frame, most of the shuttle's edge contours have moved out of the field of view. Therefore, we switch to using the model of the docking site. To get the initial position of the docking site, we again use the result of the previous frame, shown in (b). However, since we are switching models, the initial viewing parameters relative to the mark must be determined. We did this by aligning the mark with the mark on the previous frame to get the starting point shown in (c). In this figure we skip the intermediate result, due to lack of space, and show the final result in (d). The PDV for this run is $(0°, 21°, 23°, 2.36, -.34, .42)$.

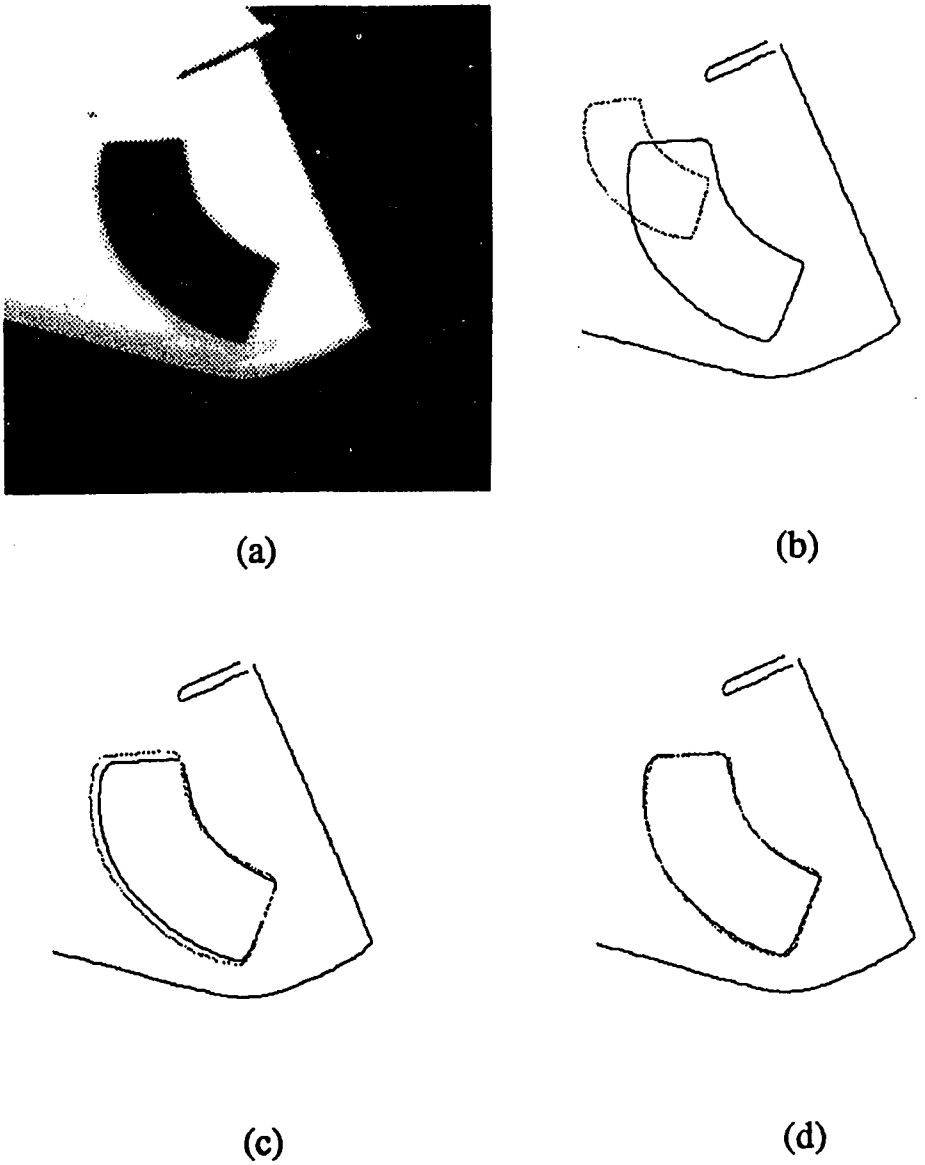


Figure 5.49. Fourth frame in a set of four in a docking scenario. Since we have approached the shuttle still more closely, we must continue to use the model of the docking site. As before, the initial pose of the model in this figure, shown in (b), is the same as the final pose in frame four, i.e., the pose shown in Fig. 5.48 (d). We have docked successfully. The PDV for this run is $(0°, 3°, 0°, 1.37, -.18, .07)$.

Figs. 5.50 and 5.51 show examples of running AEFMA on images of the shuttle that have been rotated out of the image plane. Since the shuttle is a true 3-d object, and AEFMA is using a space curve to predict the contours of the shuttle, it is not possible to achieve a perfect fit. Nevertheless, AEMFA is able to produce good estimates showing that AEFMA can operate with distorted contours.

Fig. 5.52 shows the result of running AEFMA on a cluttered image where the shuttle itself is only partially visible.

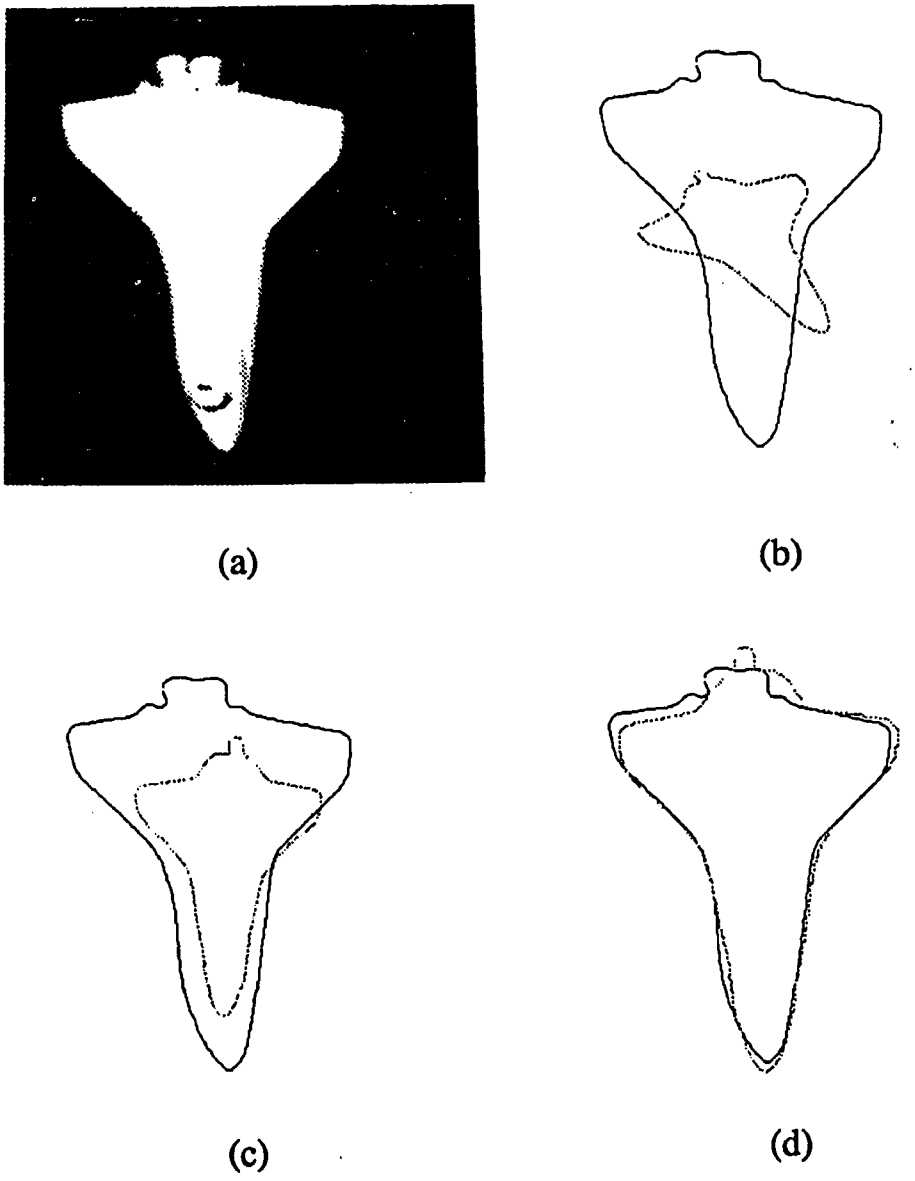
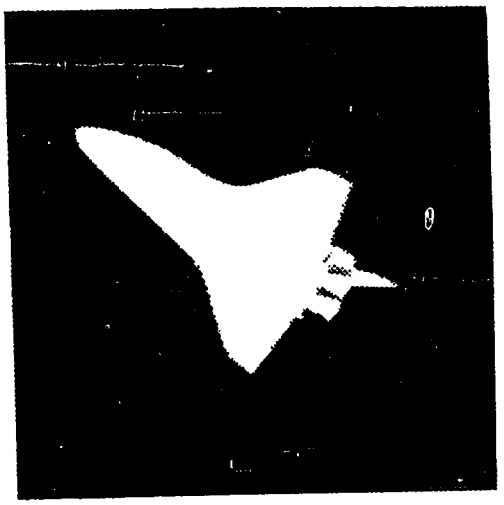
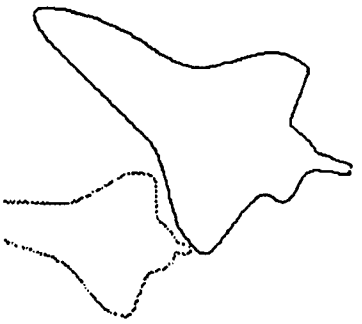


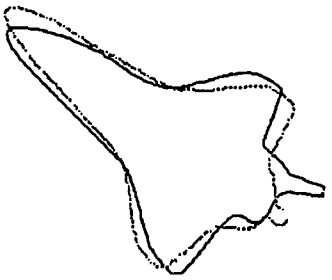
Figure 5.50. An example running AEFMA on a tilted shuttle. The PDV for this run is $(-50°, -60°, 90°, 1.52, -.04, .03)$.



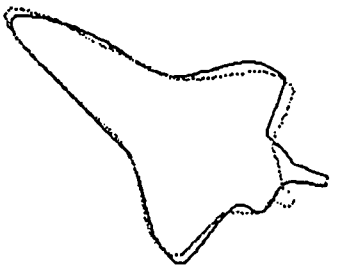
(a)



(b)



(c)



(d)

Figure 5.51. Another example running AEFMA on a tilted shuttle. The PDV for this run is $(-10°, -5°, -19°, 1.3, .40, .46)$.

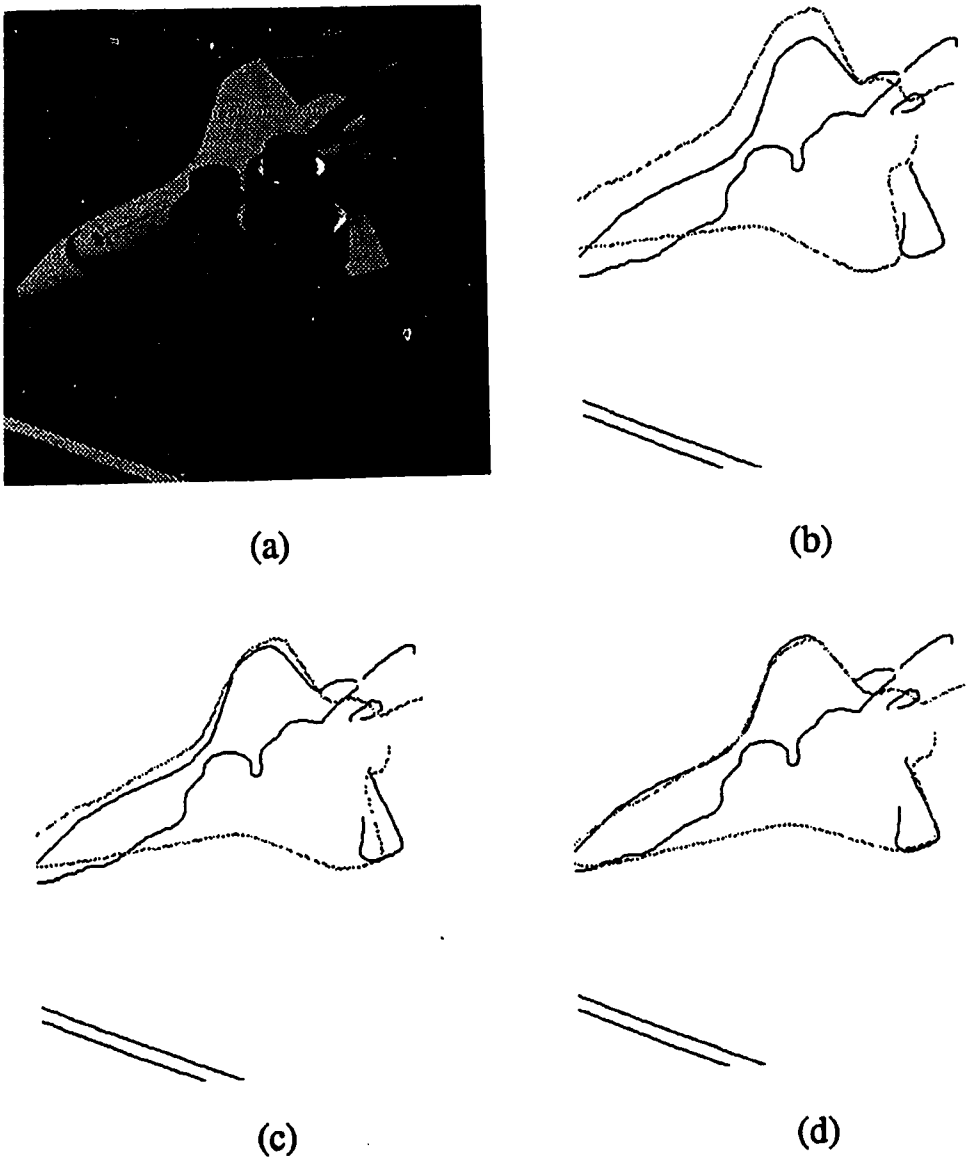


Figure 5.52. An example of running AEFMA on a cluttered image containing the shuttle.

The next experiment demonstrates tracking of the shuttle as it rotates out of the image plane. In the first frame of this sequence, shown in Fig. 5.53, the shuttle is imaged directly overhead. In the subsequent three frames, the shuttle is rotated about axes that are roughly parallel to the axis joining its wingtips. In the second frame, shown in Fig. 5.54, the shuttle has been rotated approximately ten degrees from the pose in the initial frame. In the third frame, shown in Fig. 5.55, the shuttle has been rotated approximately twenty degrees beyond the pose of the second frame. The final frame, shown in Fig. 5.56, has been rotated by more than thirty additional degrees from the pose of the third frame, a large rotation. However, as can be seen from the figures, AEFMA was able to track the object even through these large rotations. As in the previous tracking scenarios, the final pose of the current frame is used as the initial pose of the subsequent frame.

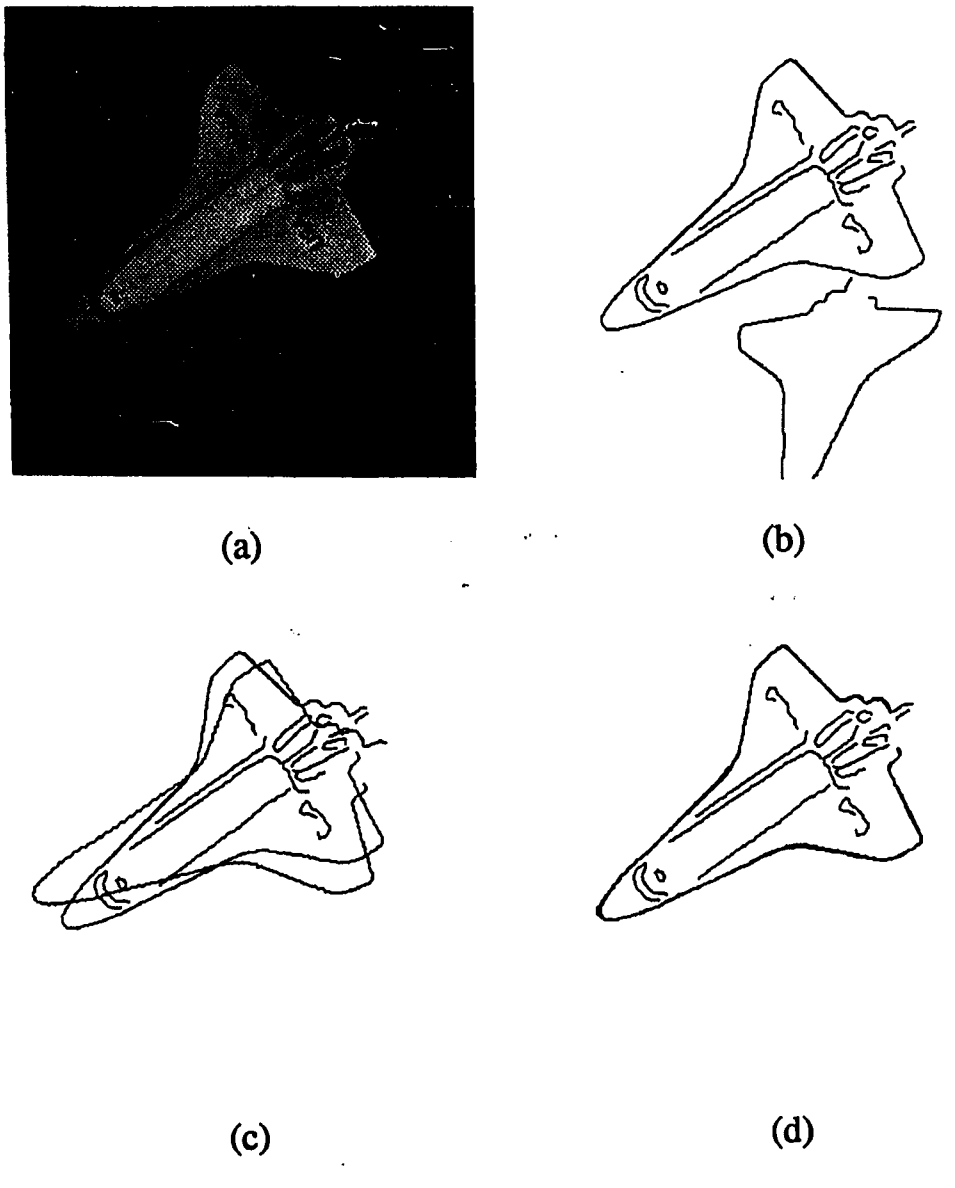
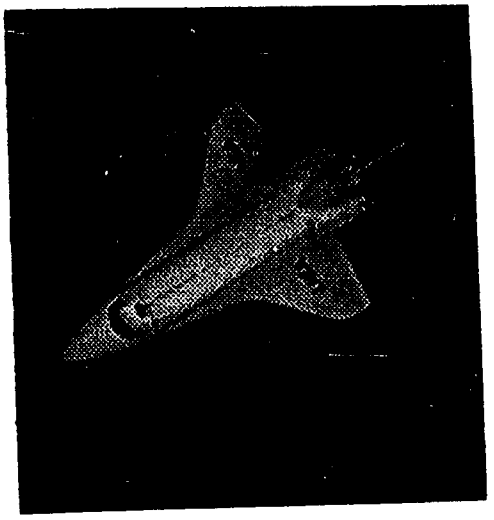
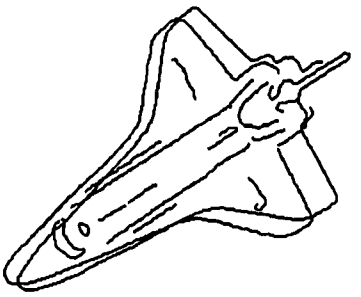


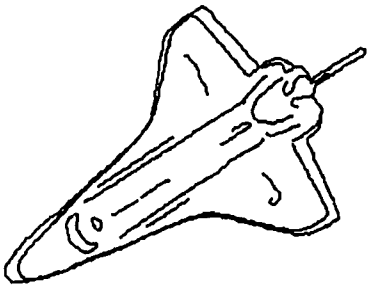
Figure 5.53. Initial frame in a set of four demonstrating tracking an object as it rotates out of the image plane. The PDV for this run is $(43°, 54°, 1°, 1.18, .18, -.64)$.



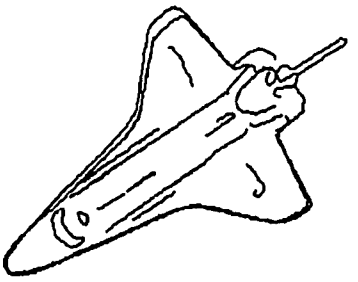
(a)



(b)



(c)



(d)

Figure 5.54. Second frame in a set of four tracking the space shuttle as it rotates out of the image plane. The PDV for this run is $(10°, 1°, -3°, .93, .05, -.04)$.

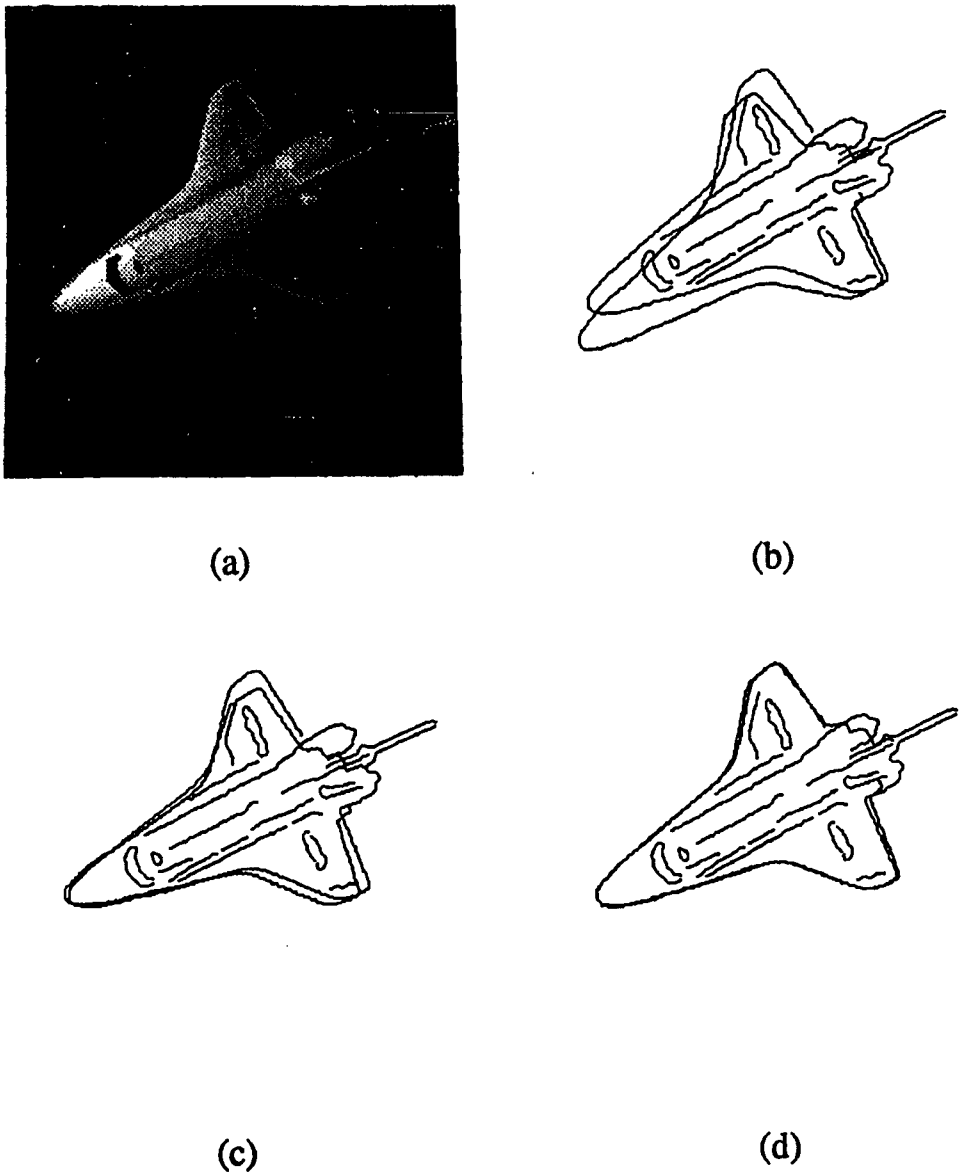
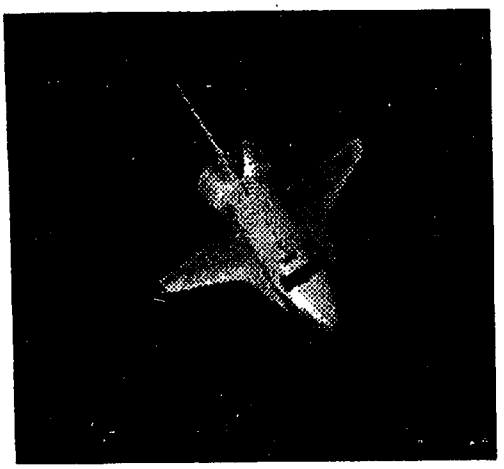
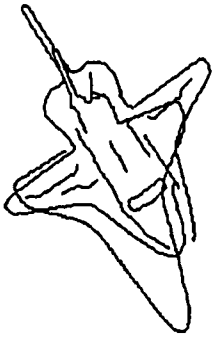


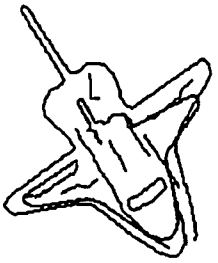
Figure 5.55. Third frame in a set of four demonstrating tracking of the shuttle as it rotates our of the image plane. The PDV for this run is $(19°, 3°, -6°, 1.08, -.03, . - .08)$.



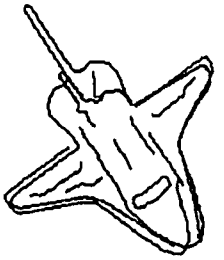
(a)



(b)



(c)



(d)

Figure 5.56. Fourth frame in a set of four demonstrating AEFMA tracking the shuttle as it rotates out of the image plane. In this frame the shuttle has been rotated by a large angle, greater than thirty degrees out of the image plane as compared with the last frame. The PDV for this run is $(33°, 3°, 4°, 1.3, .11, -.02)$.

The next experiments, shown in Figs. 5.57 and 5.58 test AEFMA's behavior on images containing multiple objects of a similar nature.

In Figs. 5.57 and Fig. 5.58 show that, although other objects of similar nature are nearby, the shape primitives are sufficiently selective that the minima of the CDF is influenced primarily by shape primitives on the correctly matching object.

The experiments above provide empirical evidence that AEFMA is a practical approach to viewing parameter estimation that could be used as part of a 3-d, model-based recognition system, such as Cyclops, or as part of a model-based tracking system. Further, since AEFMA is designed to never invoke the object-attached feature assumption, with the proper models, AEFMA can determine the pose of a large variety of objects.

The experiments also showed that AEFMA has a large range of convergence. This implies that the poses in initial hypotheses provided to AEFMA by a hypothesis generator need not be very accurate; AEFMA will find the correct pose. In the Cyclops framework, this implies that the multiview models (see Chapter 4) will not need to be densely covered with model instances to insure that the initial pose estimate is good.

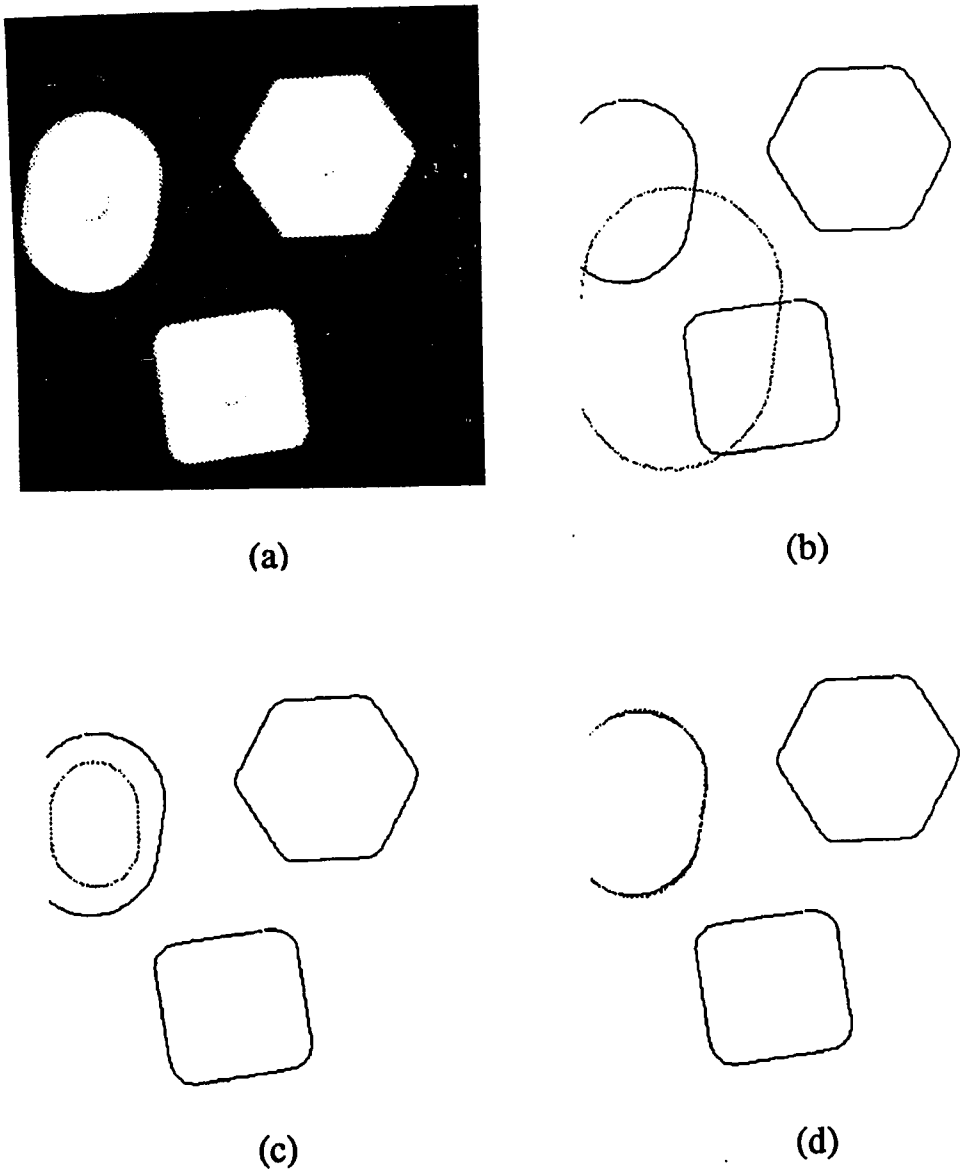
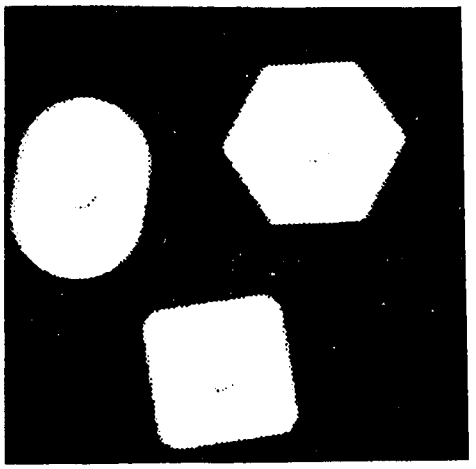
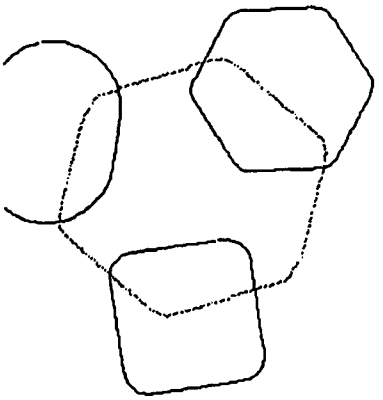


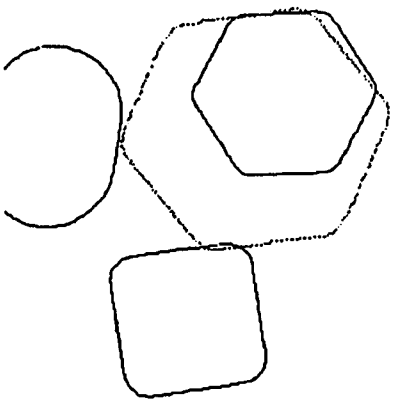
Figure 5.57. An example of running AEFMA on an image of three plastic, smoothly curving geometric figures. In this case, the model of the oblong object was used. Even though the other objects are in close proximity, AEFMA adjusts the viewing parameters to correctly place the model over the oblong object in the image. The PDV for this run is $(0°, -14°, -14°, .66, -.18, .48)$.



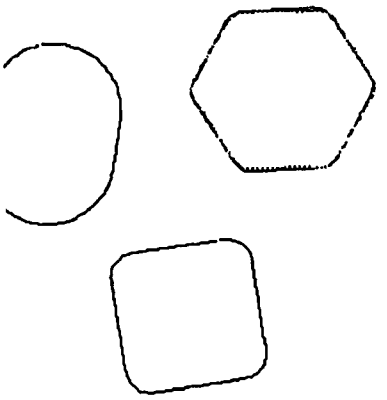
(a)



(b)



(c)



(d)

Figure 5.58 An example of running AEFMA on an image of three plastic, smoothly curving geometric figures. In this case, the model of the oblong object was used. Even though the other objects are in close proximity, AEFMA adjusts the viewing parameters to correctly place the model over the oblong object in the image. The PDV for this run is $(-5°, -9°, -8°, .66, .33, .35)$.

CHAPTER 6

CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

Throughout this thesis we have been concerned with recognition and viewing parameters estimation of 3-d objects in intensity images. We first defined the problem of recognition, and saw that estimating the viewing parameters is a key component of the recognition process.

6.1 Summary of Contributions

A major thread in this thesis has been the concept of object-attached features, the limitations imposed by using them, and performing recognition and viewing parameter estimation without using them. Indeed, in Chapter 3, we identified the *object-attached feature assumption* as a little acknowledged but pervasive part of the underpinning of many object recognition approaches. We showed how reliance on the object-attached feature assumption to simplify the problems of recognition and viewing parameter estimation limits recognition to objects that can produce such features. We saw that a large class of objects that do *not* reliably produce such features are smoothly curving objects. Thus, we resolved to examine how to perform recognition and attitude determination without assuming object-attached features.

Chapter 2 described a framework for recognition that never invokes the object-attached feature assumption. This results in a framework that is more general, but more complex than typical methods. The work in this thesis has concentrated on the parts of the

framework for existing methods were not effective in the absence of the object-attached feature assumption. Thus, Chapter 4 was concerned with *hypothesis generation*. There, we showed that disallowing object-attached features leads, by necessity, to the use of a *multiview feature representation* of the objects, where the representation consists of the 2-d features predicted to appear from the vairious views of the object. We then characterized the process of generating hypotheses explaining a particular image feature as a *neighborhood search* through all of the predicted features for those features that possess attributes similar to the image feature's attributes. In addition, we developed a method for performing neighborhood searching based on a multidimensional data structure called a k-d tree that is extremely fast, being logarithmic in the number of features. Conventional approaches to hypothesis generation are usually based on linear and, in some cases, sub-linear, searching strategies. The effectiveness of this approach was demonstrated through the experimental results obtained using a 2-d recognition system on images of partially visible puzzle pieces and micro switch parts. In the 3-d case, issues and tradeoffs between features and performance of the hypothesis generation process were analysed.

Viewing parameter estimation, of all of the parts of the framework put forward in this thesis, is perhaps the most affected by the object-attached feature assumption. In Chapters 3 and 5 we have observed that viewing parameter estimation is simple problem that results in a system of equations that, in many circumstances, has an analytic solution or can be solved with simple numerical methods. Without the object-attached feature assumption, estimating the viewing parameters becomes markedly more complicated. Chapter 5 tackled this problem. The result was an approach that we have called **A***ttitude* **E***stimation by* **F***eature* **M***odulated* **A***ttractors*, or AEFMA. This approach is based on minimizing a carefully constructed measure of the disparity between the shape of edge contours detected in the image and the edge contours predicted from the model. The disparity measure is constructed from individual disparities measured between the *shape primatives* that comprise the representation of the image edge contours

and the predicted edge contours. The selectiveness of these primatives endows the composite disparity measure with the property that it is easy to minimize using the conjugate gradient method even when the initial viewing parameters are grossly erroneous. High accuracy in the final estimates are obtained by iteratively "sharpening" the multidimensional valley forming the global minimum. In Chapter 5, the method implemented and experimentally characterized. Applications to model-based tracking were also discussed there, as were methods for quickly and accurately predicting the appearence of edge contours from any viewpoint.

6.2 Directions for Future Research

The state of object recognition is now very exciting. It is now within our grasp to create a practical 3-d recognition and tracking system using the approach of this thesis as a basis. Many issues remain to be examined. For example, hiearchical models permiting recognition tracking of objects from wide range of distances would be very useful, especially in space docking applications. This would also lend itself to recognizing and tracking articulated objects such as pliers and robot arms. The AEFMA approach is sufficiently general that, with the proper type of models, perhaps based on superquadrics [Bar81, SB90] deformable objects, or generic object classes, could be matched to an image.

There are a few more specific items. We believe that the convergence of AEFMA can be improved considerably by putting some knowledge into the minimization procedure. Accomplishing this may allow AEFMA to perform real-time tracking and attitude determination. Concerning the multiview feature representation touched on in this thesis, the optimal placement of views over the viewing sphere largely unsolved. Finally, extending the multiview feature representation to deformable models would permit recognition of deformable objects.

APPENDICES

APPENDIX A

KNOWLEDGE-BASED CONTOUR GROUPING

This appendix describes the details of the knowledge-based contour grouping approach discussed briefly in Section 4.3.2.7. As shown there, if a general recognition algorithm could be given some information about what features belong to a single object, then a significant speedup could be obtained (assuming that it takes less computation to obtain this information than it does to do without it).

A.1 Overview of the Knowledge-Based Contour Grouping Module

The processing of data starts with an image and proceeds through the phases enumerated below.

1. Edges are detected in the image using a Canny edge detector. The smallest sigma consistent with the amount noise present in the image is used. A typical value is $\sigma = 4.0$ pixels units. Three images result from the application of the Canny operator: an edge map, $e(x, y)$, where edge pixels are marked by a 255 value at an edge point and 0 elswhere, and the two gradiant components $g_x(x, y)$ and $g_y(x, y)$. The direction of the edge at a point (x_0, y_0) is given by $\arctan(g_y(x_0, y_0)/g_x(x_0, y_0))$.

2. A simple linker is applied. It operates by searching the neighboring edgels to the current one in the following manner: first the pixel most nearly in the direction indicated by the gradiant is checked. If it is marked as an edgel, the search stops

here. If this pixel is not marked, the ones on either side of it are checked, and so on, out to a user definable limit. If a marked edgel is discovered at any point, it is added to the contour, and the process is recursively continued using the newly discovered edgel as the current edgel.

3. The contours obtained in the previous stage are processed to obtain entities which we have called *hot spots*. In general terms, a hot spot is a grouping of contour endpoints wherein the member contour endpoints are near enough to each other to indicate that they *may* bear a relation to each other different from the trivial not-related case. In the process of detecting hot spots, some of the original contours may be broken, especially at points where proximity suggests that the simple linker may have made a mistake. The hope is that the knowledge based stage (following this one) will correct the mistakes of the simple linker and result in the correct linking.

4. Each pair of contour endpoints from a hot spot is given a set of fuzzy labels indicating the relation between the two contour endpoints. The system then constructs a global data structure in which the labels are the current state of the system. The contour data from the previous stages is also included. Rules, consisting of condition-action pairs, are applied to the contour endpoints at each hot spot until a termination condition is met. The rules are constructed from a set of access functions for the data structure as well as a set of fuzzy logic functions. The control structure is very simple and contains no backtracking. The process may be thought of as rule-based relaxation labeling [BB82]. The result of the rule-based module is a connectivity matrix between the contours.

The remainder of the report discusses items 3 and 4 in detail.

A.2 Hot Spot Detection

The process of detecting hot spots is centered around the existing contour terminations. Wherever any part of one contour approaches closely the termination (or endpoint) of another contour, a hot spot is formed. Formation of the hot spot may necessitate the splitting of the first contour. How close a contour should be allowed to approach before it should be (possibly) split and its endpoint(s) added to the hot spot is really a design parameter which specifies how global the linking process should be. A tradeoff exists: one would like to make the distance as small as possible so as to make the average number of contour endpoints per hot spot as small as possible for efficiency, while making it too small would cause some contours to be omitted from hot spots in which they should be included, or cause certain hot spots to not be created at all when they should. We have determined empirically that making this distance somewhat larger than the smallest meaningful features that may appear in the image works well.

A.3 Knowledge-Based Contour Grouping

The code for the knowledge based portion of the algorithm has been implemented in common lisp and C. There are three main parts to the knowledge based algorithms.

- The global data structure and its access functions.
- The rules and the fuzzy logic they employ.
- The control structure.

Contour data structure.

```
(defstruct  contour
    id-number       Id of the contour
    pointer         Pointer to the C data structure
                    containing the x-y locations and other information.
    sconns          References to the ids of contours
                    possibly connected to the start of this contour.
    econns          References to the ids of contours
                    possibly connected to the end of this contour.
)
```

Figure A.1. Contour data structure used in the contour grouping algorithm.

A.3.1 Global Data Structure

A.3.1.1 Structural Description

The global data structure (GDS) consists of a hash table of contour data structures. The central lisp data type is a contour structure defined in lisp in Fig. A.1.

Each contour is given a unique identification number which is stored in the field **id-number**. The entry **pointer** contains a pointer to the data contour The pointer references a data structure which contains all of the /xy/ coordinates of the contours (after hot spot detection), the length of the contour, and the normalized gradiant values at each edgel. The field **sconns** contains a list of all endpoints that belong to the same hot spot as the contour's start point. Similarly, the field **econns** contains a list of all endpoints that belong to the same hot spot as the contour's endpoint. Both are a lisp forms whose format is shown in Fig. A.2.

The **connrec** data type is defined by the lisp structure shown in Fig. A.3

The **connrec** data type contains the fuzzy truth values of the three possible types of connections that may exist between the start of the contour and the endpoint specified by

sconns | eccons	:=	(<endpoint specifier> <connection record>)
<endpoint specifier>	:=	(<contour id number> [start | end])
<connection record>	:=	<an instance of a **connrec** data type>

Figure A.2. Format of the contour connection connection list

Connection data structure.

```
(defstruct connrec
    (continuation .33333)    Certainty that this contour is a continuation
                             of the current contour.
    (join .33333)            Certainty that this endpoint joins the
                             current one (not as a continuation).
    (noconnection .33333)    Certainty that this endpoint has no
                             connection to the current contour.
)
```

Figure A.3. Connection label data structure.

<endpoint specifier>. As will be described more fully in the next section, all fuzzy truth values range from 0 to 1. The **continuation, join,** and **noconnection** fields are disjoint and so we have constrained them to sum to unity in a manner similar to probabilities of disjoint events covering a sample space. Since nothing is known at the start about the relations between the endpoints, the fields of the **connrec** data type are initialized to 1/3.

The econns field of the contour data structure is the same as the sconns, except that it refers to connections to the end of the contour (the start and end of the contour are unambiguously defined by the C data structure).

Fig. A.4 gives an example of a typical contour data structure after the system has been allowed to run some time. The entry under **sconns** implies that the system has decided with certainty that the end of contour 12 is a continuation of the start of contour 9. Similarly, the entries under **econns** say that the system gives the start of contour 10 is a continuation with certainty of approximately .2, a join with certainty of approximately .8, and no chance of a noconnection while the start of contour 11 is a join with complete certainty.

```
#S(
      contour id-number 9

      pointer 242400

      sconns
      (((12 end) #S(connrec continuation 1.0 join 0.0 noconnection 0.0)))

      econns
      (((10 start) #S(connrec continuation 0.207 join 0.792 noconnection 0.0))
      ((11 start) #S(connrec continuation 0.0 join 1.0 noconnection 0.0)))
)
```

Figure A.4. Example of a contour data structure after the grouping module has been allowed to run for a while. The #S is the lisp print symbol for a defined data type.

The contour structures for all contours found in an image are placed in a lisp hash table with the contour id numbers as the hash keys so that they may be accessed quickly. This hash table (as will be discussed more fully later, two copies of it exist) is the entire GDS or black board of the system.

A.3.1.2 Access Functions for the GDS

The access functions can be divided into two classes: those which read and change the certainty values of the labels and those which return information about the contours themselves. Below some of the more important routines and their functions are mentioned. Typically the routines take one or more endpoint specifiers as their inputs since, as we will discuss later, the rules are written in a form that expects the variables in them to be bound to endpoint specifiers.

The first set of access functions deal only with reading or changing the fuzzy certainty values of the labels. All of the functions that change a label insure that the unity summation constraint is obeyed, as well as insuring that the labels remain in the range from 0 to 1. There are actually two copies of the GDS around at any time: the one that has the current state, called the current blackboard or cbb, and the one that is modified

based on the current state, called the next black board or nbb. These will be discussed more in Section A.3.3 on the control algorithm. Functions that inquire the blackboard's current state do so to the cbb while functions that modify the state do so to the nbb. In addition, the access functions were designed to operate symmetrically if more than one endpoint are given as arguments.

- **(set-values** *endpoint1 endpoint2 cval jval nval***)**

  This function sets the connrec entries of the named endpoints to *cval*, *jval*, and *nval* respectively, scaling them proportionately if they do not satisfy the unity summation constraint. If there is no connrec entry for *endpoint1* and *endpoint2*, the call returns **nil** and does nothing, otherwise it returns **t**.

- **(getcval** *endpoint1 endpoint2***)**

  (getjval *endpoint1 endpoint2***)**

  (getnval *endpoint1 endpoint2***)**

  The function **getcval** returns the certainty of a continuation existing between *endpoint1* and *endpoint2*. Similarly for **getjval** and **getnval** except that these operate on **join** and **noconnection** labels respectively.

- **(incr-cval** *endpoint1 endpoint2 incrfactor***)**

  (incr-jval *endpoint1 endpoint2 incrfactor***)**

  (incr-nval *endpoint1 endpoint2 incrfactor***)**

  The function **incr-cval** increases the certainty factor of a continuation relation existing between *endpoint1* and *endpoint2* such that the difference between the current value and 1.0 is reduced by the fraction specified by *incrfactor*. The other relations are reduced proportionately to maintain unity summation. The analogous functions for joins and noconnections are **incr-jval** and **incr-nval** respectively.

- **(decr-cval** *endpoint1 endpoint2 decrfactor***)**

  (decr-jval *endpoint1 endpoint2 decrfactor***)**

(decr-nval *endpoint1 endpoint2 decrfactor*)

These functions are exactly analogous to the **incr-cval, incr-jval,** and **incr-nval** counterparts described above except that they reduce the current value according to the formula $new = (1 - decrfactor)current$, where *new* is the new label value, and *current* is the current label value.

- **(zero** *endpoint1 endpoint2 whichval*)

  This function zeroes the certainty relation specified by *whichval* (i.e. **continuation, join,** or **noconnection**) and proportionately scales the other certainties to abide by the unity summation constraint.

- **(nconns** *endpoint*)

  Returns the number of other contours that are a member of the hot spot that *endpoint* is a member of.

- **(epid** *endpoint*)

  This function simply returns the id number of the contour specified by *endpoint.*

The second set of access functions are those that compute quantities from the contour data. This data is referenced via the pointer that is stored in one of the fields of the lisp contour data type. Perhaps one of the most important of these functions is one which fits a line to and endpoint of a contour. I have used a weighted least squares line fit (also known as a chi-square line fit) [PFTV86] to fit lines to the ends of the contours. The weight of the endpoint is the largest and the weights decay monotonically to zero as the distance from the endpoint of the contour is increased. The function **fit-line-to-endpoint** returns the fitted lines in the lisp form $((i_x\ i_y)\ (p_x\ p_y))$ where i_x and i_y are the coordinates of the unit vector in the direction of the line, and p_x and p_y are the coordinates of the point nearest to the endpoint of the contour on the line.

Below are the remainder of the important access functions.

- **(set-weights** *sigma*)

  This function sets the width of the weights in the chi-square line fit.

- **(fit-line-to-endpoint** *endpoint*)

  As described above, this function performs a chi-square line fit to the contour data at *endpoint* using a Gaussian to weight the data. The lines are returned in the form described above.

- **(endpoint-location** *endpoint*)

  Returns the coordinates of the specified endpoint in the lisp form (x y).

- **(angle** *endpoint1 endpoint2*)

  Returns the angle between the linear extrapolation of endpoint1 and endpoint2 as performed by **fit-line-to-endpoint**. The angle is in the range $[0, \pi]$.

- **(contour-length** *contouridnumber*)

  Returns the length of the contour in pixel units.

The last set of functions are utility functions used frequently in the rules.

- **(dist-point-to-line** *line point*)

  This function takes a line of the form described above and a point in the form (x y) and computes the distance from the point to the line in pixel units.

- **(dist** *point1 point2*)

  Returns the distance between the points in pixel units.

- **(intersect-lines** *line1 line2*)

  Returns the point of intersection between the lines.

A.3.2 Fuzzy Logic

All the rules in the contour grouping module are based on fuzzy logic. Fuzzy logic allows much more precision in the specification of rules. The fuzzy logic that we have used is a mixture of that described by Ohta [Oht85] and that used in the MYCIN system [DBS77]. but is more similar to MYCIN than to Ohta's work. However, we have kept the logic in a form that affords its interpretation as a probability in situations where it is useful to do so. We explain the details of the fuzzy logic below.

- **Fuzzy Logic Values**
  0 represents no evidence.
  .5 represents some evidence.
  1 represents complete evidence.

- **Fuzzy Logic Functions**
  Let f_1 and f_2 be fuzzy logic values. Let b be a Boolean logic value (i.e. true or false) Then:
  (andf f_1 f_2**)** is defined as $\min(f_1, f_2)$.
  (orf f_1 f_2**)** is defined as $\max(f_1, f_2)$.
  (notf f_1**)** is defined as $1 - f_1$.
  (boolf b**)** – 1.0 if b is true, 0.0 otherwise.

The labels, **join**, **continuation**, and **noconnection**, are mutually exclusive. Therefore, we have taken advantage of the probabilistic interpretation of the certainties to require that the sum of the certainty values of the labels is unity, as would be the case for disjoint events covering a probability space.

In order to convert the numerical output of the access functions into fuzzy truth values, a number of conversion functions are used. My conversion functions are shown graphically in Fig. A.5 (a), (b), (c), and (d). They are called *lp()*, *hp()*, *bp()*, and *notch()* respectively. These names stand for *low pass*, *high pass*, *band pass*, and *notch* respectively. We have used this terminlogy because of the resemblance the conversion functions bear to the frequency domain characteristics of the corresponding filters.

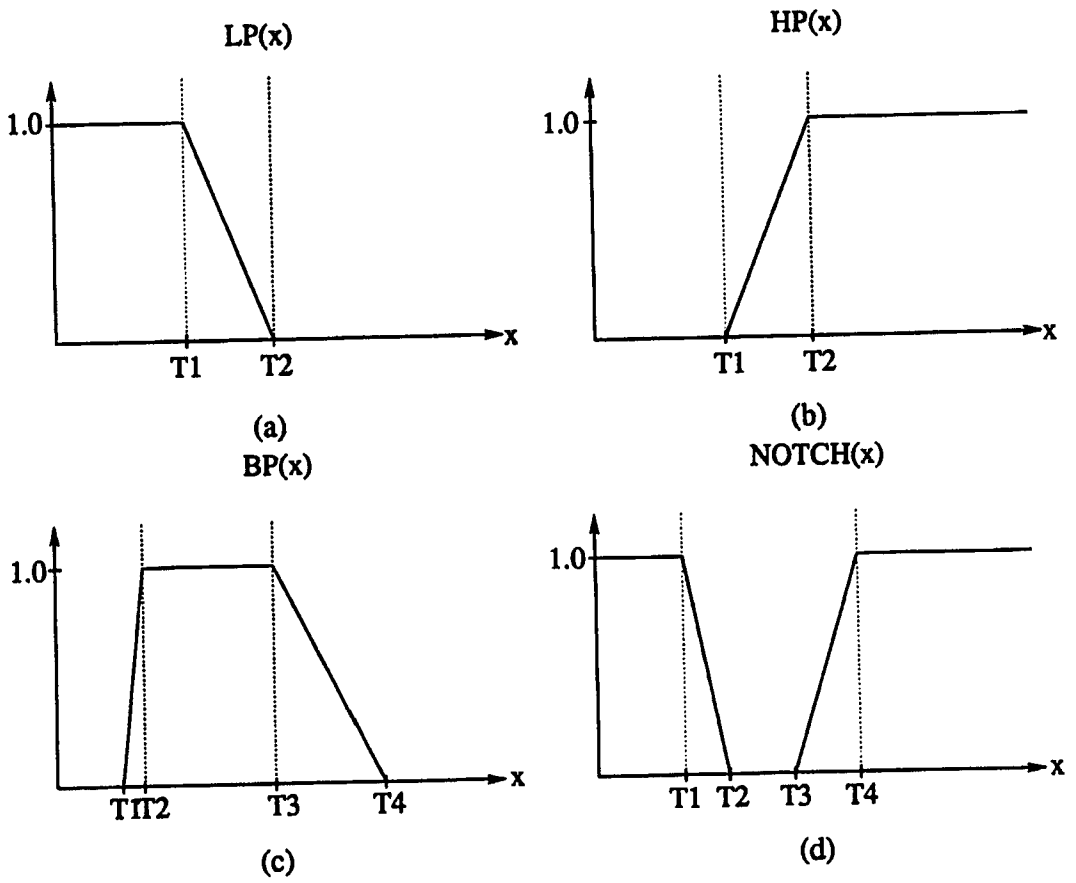


Figure A.5. Shown above are the functions which convert numerical values to fuzzy truth values. The functions each have several parameters which are also shown in the figure.

A.3.3 Rules and Control

A.3.3.1 Structure of Rules

The rules are lisp forms of the following structure:

<rule> := (<rule number> <condition> <implication strength> <action>)

The rule number is a unique, non-negative integer identifying the rule.

The condition part is an executable piece of lisp code. All variables appearing in the condition are of the form x_n, where n is an integer, and they are bound to arbitrary

```
(andf   (andf   (boolf (< 1 (nconns x0))) (hp (angle x0 x1) 120 160))
        (lp     (+
                        (dist-point-to-line (fit-line-to-endpoint x0) (endpoint-location x1))
                        (dist-point-to-line (fit-line-to-endpoint x1) (endpoint-location x0)))
                        2.0 4.0
        )
)
```

Figure A.6. An example rule.

endpoints. If more than one variable appears in the rule, the control module will bind only those endpoints which belong to the same hot spot. The control algorithm binds the x_n prior to execution of the rule. Typically, the condition part is made up of access functions and conversion functions. The "hard" part of writing a predicate for a rule consists primarily of picking good values for the parameters of the conversion functions. An example of a predicate is given in Fig. A.6.

This predicate will be true if the number of connections to endpoint x_0 is greater than 1, and the angle between endpoints x_0 and x_1 is more than about 140° and the sum of the distances from the extrapolations from the endpoints x_0 and x_1 to the other endpoint, x_1 and x_0 respectively, are less than about 3 pixels.

The implication strength is a number in the range $[0, 1]$ which tells how strongly the truth of the predicate should be used in the action. How this is accomplished will be described in the next section addressing the execution of rules.

The action part of the rule is also an executable lisp form in which all variables are bound to the same end points as the last execution of a predicate.

A.3.3.2 Execution of Rules and Control

The set of rules are organized into a list of lists. The inner lists group all rules that have the same number of variables appearing in them. For example, there may be a list of unary rules which have only x_0 appearing in them, binary rules which have only x_0

and x_1 appearing in them, and tertiary rules which have x_0, x_1 and x_2 appearing and so on. The outer list is just a list of these lists. The control module binds the x_i in the following manner: x_0 may be bound to any endpoint. The following variables are bound to only endpoints that are connected to x_0 (i.e. they are members of the same hot spot as x_0) such that the tuple $(x_0, x_1, \ldots, x_n)$, where n is the largest number of variables appearing in any rule, is the only permutation of those endpoints that will be bound and such that none of the x_i will be bound to the same endpoint. After the binding occurs, the control module executes the predicate and compares the result with a global variable **PTHRESH***. If it is greater than **PTHRESH*** the value of the predicate is bound to a global **PVAL*** and the product of the implication strength and **PVAL*** is bound to the global **PROD***. The value of **PROD*** gives a measure of how strong the action a rule will take should be. The purpose of binding these quantities with globals is to allow the action parts of the rules to use the information about the strength of the predicate and the strength of the implication in the rule.

After the globals have been bound, the control module executes the action part of the rule. The action part of the rule often will reference the global **PROD*** to modify the blackboard proportionately to its value.

The system is set up so that the execution of the rules occurs in parallel. This allows all rules that fire to have their actions applied to the blackboard and thus eliminates the need for conflict resolution. This is accomplished by keeping two copies of the blackboard: the current blackboard (cbb) and the next blackboard (nbb). As was mentioned previously, the access functions that return information about the state of the black board inquire the cbb, while the functions that change the state of the black board do so to the nbb. Thus the action of one rule will not affect the firing of another rule and so, to the extent that the effects of the actions of the rules are commutative, the order of firing is irrelevant. The action parts of the rules are not exactly commutative, however they are approximately commutative in most cases. An iteration of the system occurs when all of the possible

combinations of the variables have been bound, and the rules applied to them. At this time, the control portion of the system copies the data contained in the nbb to the cbb and starts a new iteration.

In the ideal system, termination of processing should occur when the state of the blackboard does not change significantly after an iteration. We did not build in a check for such a condition. Currently, the action parts of the rules will set a global flag if they change the blackboard. They will not change the blackboard if the predicate does not fire or if the label certainties are 0.0 or 1.0. In many cases, the certainty of labels converges to 0.0 or 1.0 (and we have some rules which set them to 0.0 or 1.0 if the value approaches too near to either value). However, occasionally, the certainty converges to a number between 0 and 1. In this case, the program, as currently implemented, would not terminate except by outside intervention. To handle this case, a check of whether the blackboard has changed is needed.

A.3.3.3 Examples of Rules

In this section we show in detail the construction of some rules. We will not go through all of them as they are easily understood after a few examples have been seen.

One of the more interesting (and lengthly) rules is rule number 310. The lisp code for rule 310, a tertiary rule, is given in Fig. A.7. In english, this rule states:

> If three endpoints have the property that, taken pairwise, none of them are have strong continuation labels, and the sum of the distance between the points of intersection of pairs of extrapolated lines is less than about 7.0 pixel units, then increase the join labels between the three endpoints by an amount proportional to **PROD***.

Another interesting tertiary rule is given in Fig. A.8.

This rule says:

```
(310
(andf    (andf    (notf (getcval x0 x1)) (notf (getcval x0 x2)))
         (andf    (notf (getcval x1 x2))
                  (lp  (+  (dist      (intersect-lines
                                      (fit-line-to-endpoint x0)
                                               (fit-line-to-endpoint x1)
                                      )
                                      (intersect-lines
                                      (fit-line-to-endpoint x0)
                                               (fit-line-to-endpoint x2)
                                      )
                           )
                           (dist      (intersect-lines
                                               (fit-line-to-endpoint x0)
                                               (fit-line-to-endpoint x1)
                                      )
                                      (intersect-lines
                                               (fit-line-to-endpoint x1)
                                               (fit-line-to-endpoint x2)
                                      )
                           )
                           (dist      (intersect-lines
                                               (fit-line-to-endpoint x1)
                                               (fit-line-to-endpoint x2)
                                      )
                                      (intersect-lines
                                               (fit-line-to-endpoint x0)
                                               (fit-line-to-endpoint x2)
                                      )
                           )
                  )
                  5.0 9.0
                  )
         )
)
.85
((incr-jval x0 x1 prod*) (incr-jval x0 x2 prod*) (incr-jval x1 x2 prod*)))
```

Figure A.7. An example of a tertiary rule.

```
(320
(getcval x0 x1)
.9
((incr-nval x0 x2 prod*) (incr-nval x1 x2 prod*)))
```

Figure A.8. Another example rule

If two contour endpoints have a strong continuation relation between them, then it is unlikely that a third contour will have any label but noconnection between it and the other two endpoints.

Three contours that have this kind of label configuration on the relations between them are what we normally think of as "tee" junctions.

There several more rules, however, due to space limitatations, we cannot include them all here.

APPENDIX B

ADDITIONAL 2-D RECOGNITION RESULTS

This appendix provides a number of addtional results of running the 2-d recogntion algorithm described in Section 4.5.6.3. Please refer there for details of the algorithm.

As in Figs 4.41 and 4.43, the figures in this appendix show the result of edge detection on the image on the left, and show the edge image superposed with pattern-filled final hypotheses on the right. As in Section 4.5.6.3, we divide the results into a group of experiments on images of overlapping puzzle pieces and a group of experiments on images overlapping parts of a switch assembly.

B.1 Results of Experiments on Images of Overlapping Puzzle Pieces

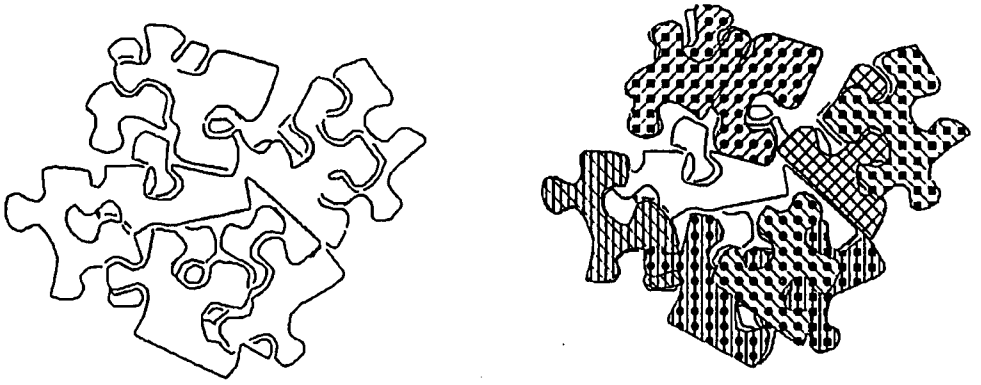


Figure B.1. Result of running the 2-d recognition algorithm on an image of overlapping puzzle pieces.

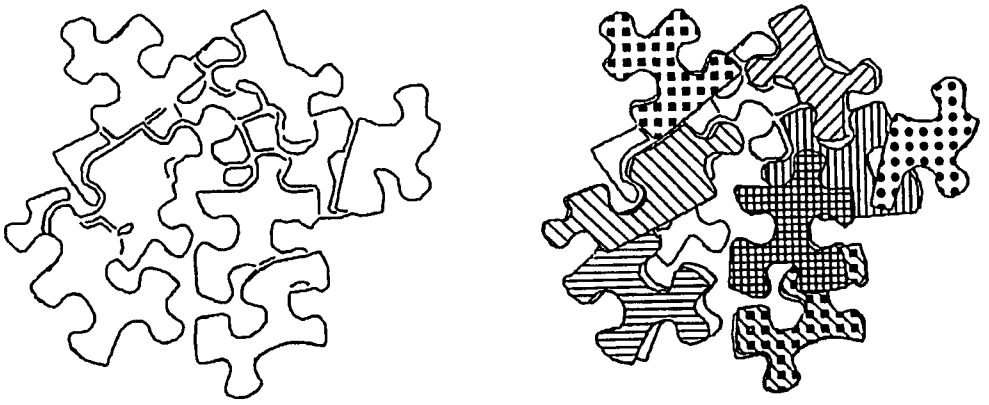


Figure B.2. Result of running the 2-d recognition algorithm on an image of overlapping puzzle pieces.

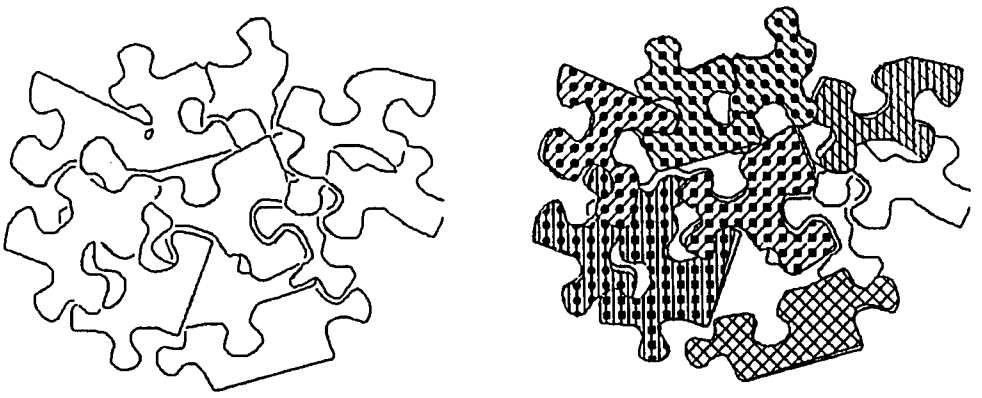


Figure B.3. Result of running the 2-d recognition algorithm on an image of overlapping puzzle pieces.

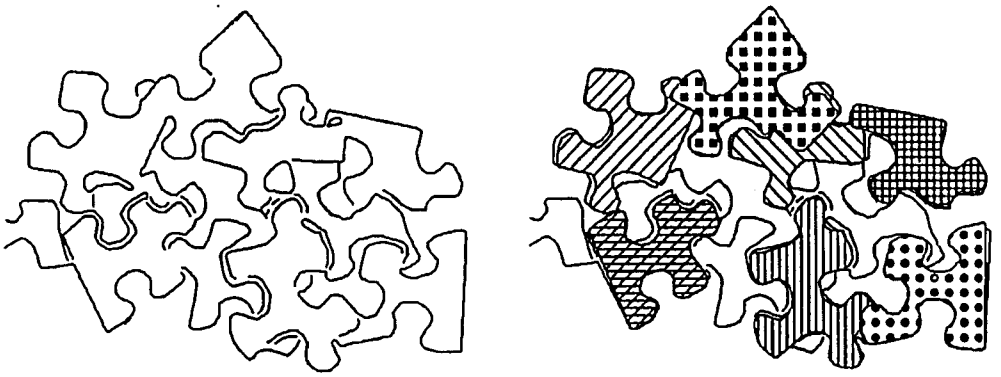


Figure B.4. Result of running the 2-d recognition algorithm on an image of overlapping puzzle pieces.

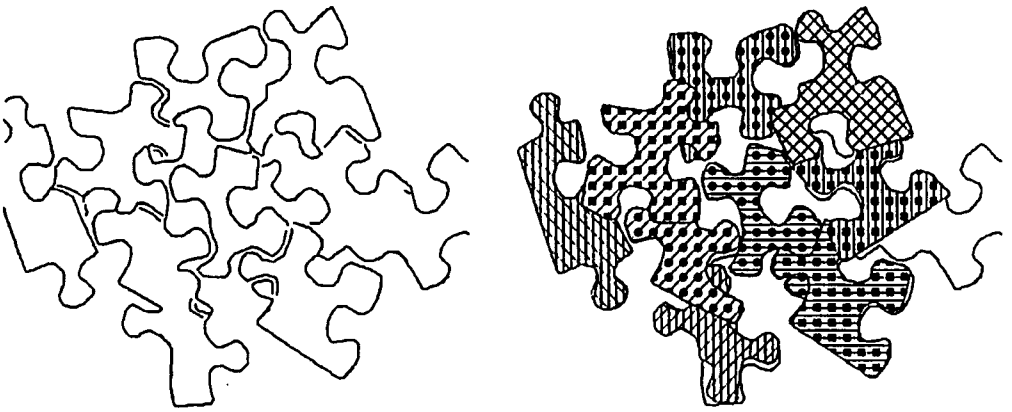


Figure B.5. Result of running the 2-d recognition algorithm on an image of overlapping puzzle pieces.

B.2 Results of Experiments on Images of Overlapping Parts of a Switch Assembly

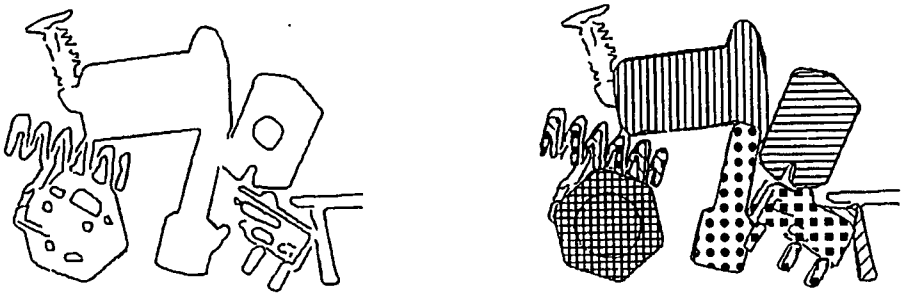


Figure B.6. Result of running the 2-d recognition algorithm on an image of overlapping parts of a switch assembly.

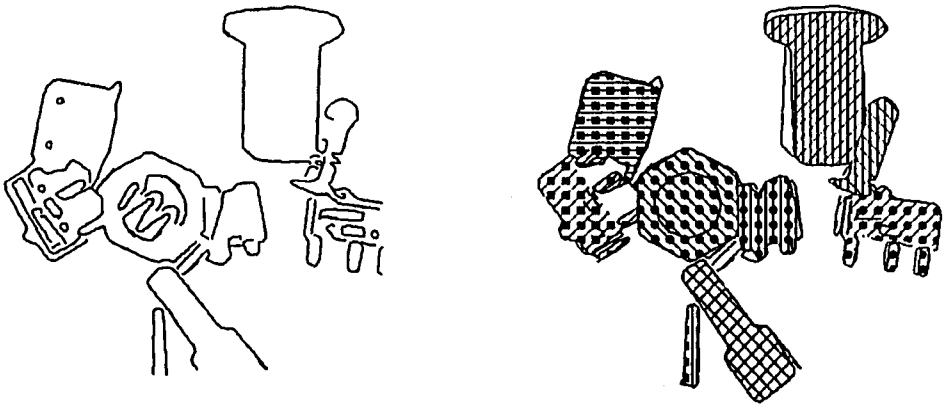


Figure B.7. Result of running the 2-d recognition algorithm on an image of overlapping parts of a switch assembly.

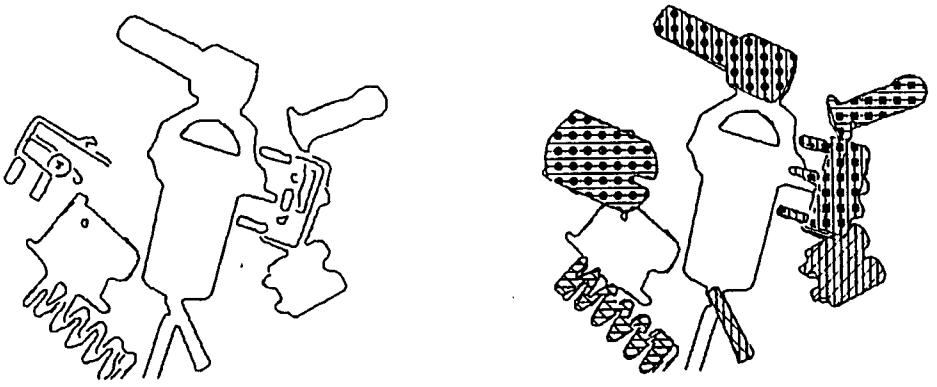


Figure B.8. Result of running the 2-d recognition algorithm on an image of overlapping parts of a switch assembly.

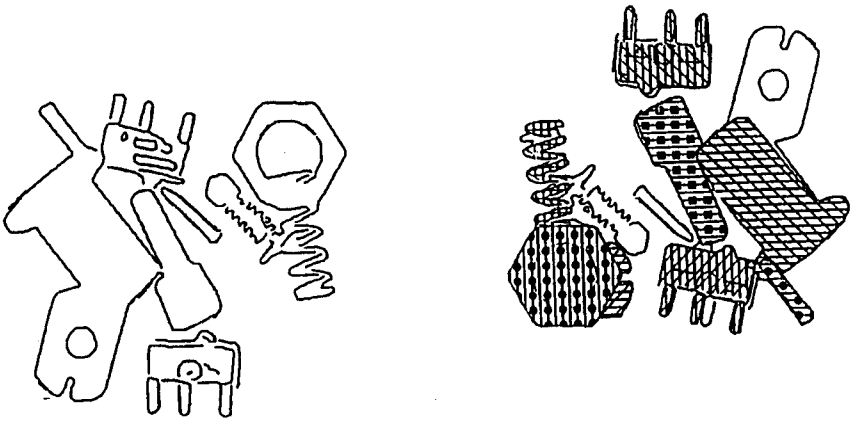


Figure B.9. Result of running the 2-d recognition algorithm on an image of overlapping parts of a switch assembly.

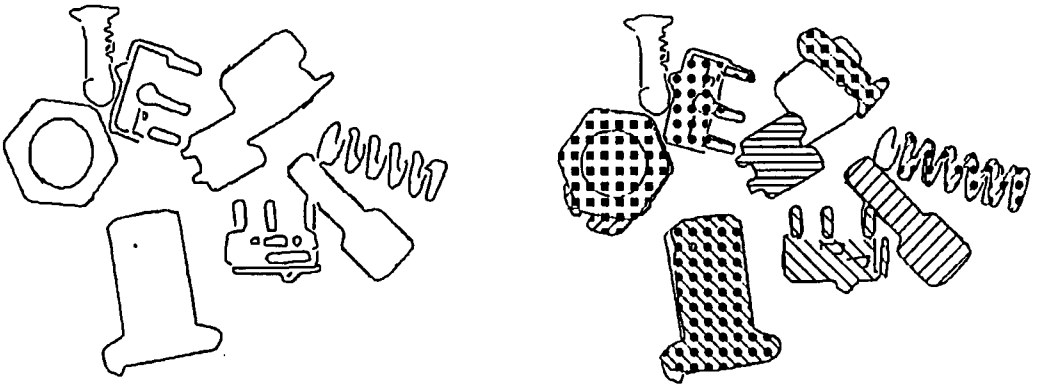


Figure B.10. Result of running the 2-d recognition algorithm on an image of overlapping parts of a switch assembly.

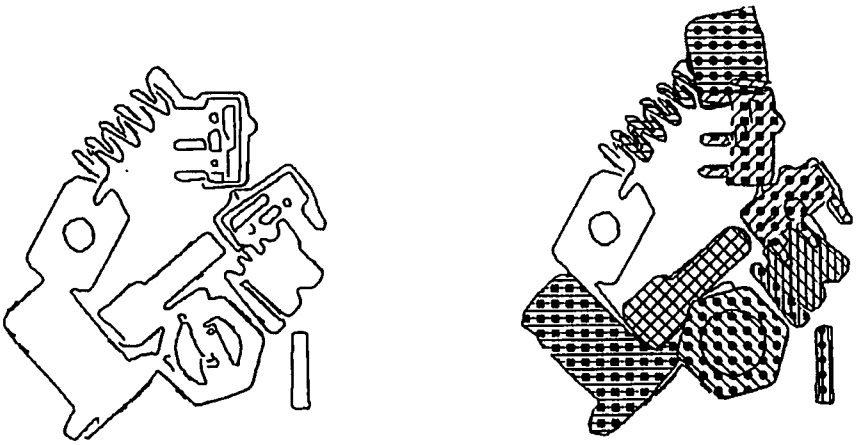


Figure B.11. Result of running the 2-d recognition algorithm on an image of overlapping parts of a switch assembly.

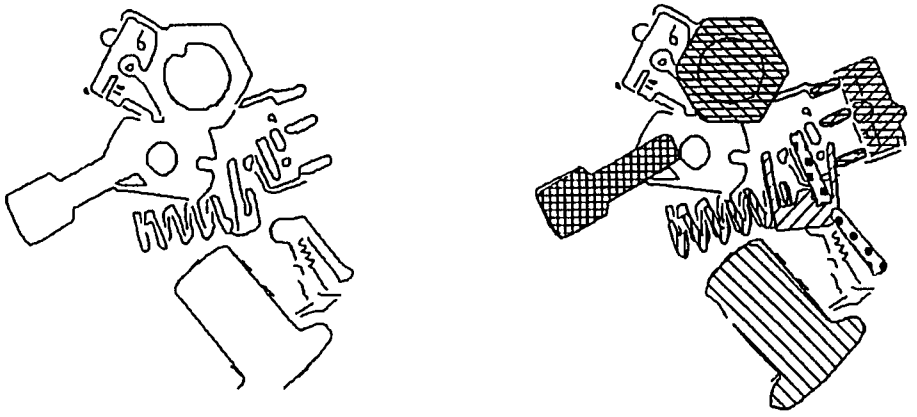


Figure B.12. Result of running the 2-d recognition algorithm on an image of overlapping parts of a switch assembly.

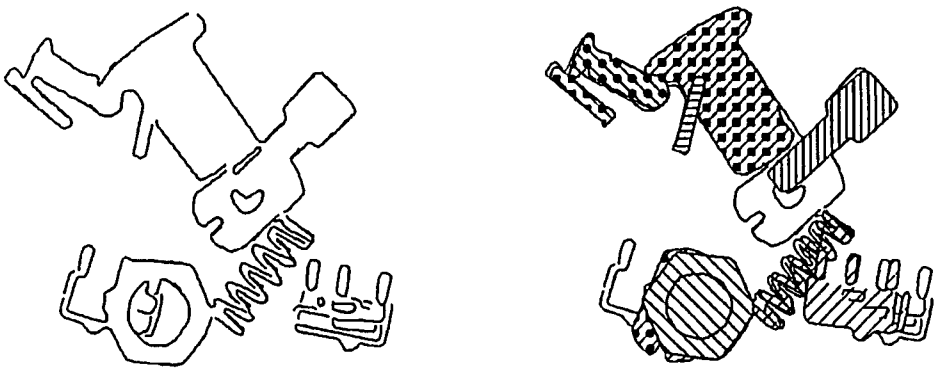


Figure B.13. Result of running the 2-d recognition algorithm on an image of overlapping parts of a switch assembly.

BIBLIOGRAPHY

BIBLIOGRAPHY

[AB86] H. Asada and M. Brady, "The curvature primal sketch," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, pp. 2–14, January 1986.

[AF86] N. Ayache and O. D. Faugeras, "HYPER: A new approach for the recognition and positioning of two-dimensional objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, pp. 44–54, January 1986.

[AFF84] N. Ayache, O. Faugeras, and B. Faverjon, "A geometric matcher for recognizing and positioning 3-d rigid objects," in *Proceedings of the SPIE Conference Intelligent Robots and Computer Vision (Vol. 521)*, pp. 152–159, 1984.

[AFFT85] N. Ayache, O. Faugeras, B. Faverjon, and G. Toscani, "Matching depth maps obtained by passive stereo," in *IEEE International Conference on Robotics and Automation*, pp. presented, not in proceedings, 1985.

[Ami90] A. A. Amini, *Using Dynamic Programming for Solving Variational Problems in Vision: Applications Involving Deformable Models for Contours and Surfaces.* PhD thesis, The University of Michigan, 1990.

[AMP84] Y. S. Abu-Mostafa and D. Psaltis, "Recognitive aspects of moment invariants," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, pp. 698–706, November 1984.

[AMP85] Y. S. Abu-Mostafa and D. Psaltis, "Image normalization by complex moments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 7, pp. 46–55, January 1985.

[Att54] F. Attneave, "Some informational aspects of visual perception," *Psychology Review*, vol. 61, pp. 183–193, 1954.

[ATW88] A. A. Amini, S. Tehrani, and T. E. Weymouth, "Using dynamic programming for minimizing the energy of active contours in the presence of hard constraints," in *Proceedings of the International Conference on Computer Vision*, pp. 95–98, IEEE, 1988.

[AWJed] A. A. Amini, T. E. Weymouth, and R. C. Jain, "Using dynamic programming for solving variational problems in vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, To be published.

[Bal81] D. G. Ballard, "Generalizing the hough transform to detect arbitrary shapes," *Pattern Recognition*, vol. 13, no. 2, pp. 111–122, 1981.

[BAM86] J. Ben-Arie and Z. A. Mieri, "3-d objects recognition by state space search: Optimal geometric matching," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recgnition*, pp. 456–461, 1986.

[Bar81] A. H. Barr, "Superquadrics and angle-preserving transformations," *IEEE Computer Graphics Applications*, vol. 1, pp. 11–23, 1981.

[Bar84] A. H. Barr, "Global and local deformations of solid primatives," *Computer Graphics*, vol. 18, no. 3, pp. 21–30, 1984.

[BB82] D. H. Ballard and C. M. Brown, *Computer Vision*. New Jersey: Prentice-Hall, 1982.

[BBR83] A. H. Bond, R. S. Brown, and C. R. Rowbury, "The effect of evironmental variation upon the performance of a second generation industrial vision system," in *Proceedings of the Cambridge Symposium on Intelligent Robots and Computer Vision*, SPIE, November 1983.

[BC82] R. C. Bolles and R. A. Cain, "Recognizing and locating partially visible objects: The local-feature-focus method," *International Journal of Robotics Research*, vol. 1, pp. 57–82, Fall 1982.

[Bel57] R. Bellman, *Dynamic Programming*. Princeton, New Jersey: Princeton University Press, 1957.

[Ben75] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[BF79] J. L. Bentley and J. H. Friedman, "Data structures for range searching," *Computing Surveys*, vol. 11, pp. 398–409, December 1979.

[BF86] B. Bamieh and R. J. P. D. Figueiredo, "A general moment-invariants/attributed-graph method for three-dimensional object recognition from a single image," *IEEE Journal of Robotics and Automation*, vol. RA-2, pp. 31–41, March 1986.

[BGB79] R. A. Brooks, R. Greiner, and T. O. Binford, "The ACRONYM model-based vision system," in *Proceedings of the International Joint Conference on AI*, (Tokyo, Japan), pp. 105–113, 1979.

[BH86] R. C. Bolles and P. Horaud, "3DPO: A three-dimensional part orientation system," *International Journal of Robotics Research*, vol. 5, pp. 3–26, Fall 1986.

[Bha84] B. Bhanu, "Representation and shape matching of 3-d objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-6, May 1984.

[BHH83] R. C. Bolles, P. Horaud, and M. J. Hannah, "3DPO: A three-dimensional part orientation system," in *Proceedings of the 8th IJCAI*, pp. 355–359, 1983.

[BM79] J. L. Bentley and H. A. Maurer, "A note on euclidean near neighbor searching in the plane," *Information Processing Letters*, vol. 8, pp. 133–136, March 1979.

[BN78] H. Blum and R. N. Nagel, "Shape description using weighted symmetric axis features," *Pattern Recognition*, vol. 10, pp. 167–180, 1978.

[Bou87] T. Boult, "What is regular in regularization," in *Proceedings of the International Conference on Computer Vision*, pp. 457–462, IEEE, 1987.

[Bre65] J. E. Bressenham, "Algorithm for computer control of digital plotters," *IBM Systems Journal*, vol. 4, no. 1, pp. 25–30, 1965.

[Bro81] R. A. Brooks, "Symbolic reasoning among 3-d models and 2-d images," *Artificial Intelligence*, vol. 17, pp. 285–349, 1981.

[Bro83] R. A. Brooks, "Model-based three-dimensional interpretations of two-dimensional images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-5, pp. 105–113, March 1983.

[BS85] D. H. Ballard and D. Sabbah, "Viewer independent shape recognition," in *Conference on Pattern Recognition and Image Processing*, pp. 67–71, IEEE, 1985.

[BU88] R. Basri and S. Ullman, "The alignment of objects with smooth surfaces," in *Proceedings of the International Conference on Computer Vision*, pp. 482–488, IEEE, 1988.

[CA87] C. H. Chien and J. K. Aggarwal, "Shape recognition from single silhouettes," in *First International Conference on Computer Vision*, pp. 481–490, IEEE, 1987.

[Can83] J. F. Canny, "Finding edges and lines in images," Master's thesis, MIT, 1983.

[Cas88a] T. A. Cass, "Parallel computation in model-based recogntion," Master's thesis, MIT, May 1988.

[Cas88b] T. A. Cass, "A robust parallel implementation of 2d model-based recognition," in *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pp. 879–884, IEEE, 1988.

[CCL84] C. K. Cowan, D. M. Chelberg, and H. S. Lim, "ACRONYM model-based vision in the intelligent task automation project," in *First IEEE Conference on AI Applications*, pp. 105–113, IEEE, 1984.

[CD86] R. T. Chin and C. R. Dyer, "Model-based recognition in robot vision," *Computing Surveys*, vol. 18, pp. 67–1083, March 1986.

[Chi89] C.-H. Chien, "Model reconstruction and shape recognition from occluding contours," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, pp. 372–389, April 1989.

[CMST88] C. I. Connolly, J. L. Mundy, J. R. Stenstrom, and D. W. Thompson, "Matching from 3-d range models into 2-d intensity scenes," in *Proceedings of the International Conference on Computer Vision*, pp. 65–72, IEEE, 1988.

[CS84] L. A. Cooper and R. N. Shepard, "Turning something over in the mind," *Scientific American*, vol. 251, pp. 106–114, December 1984.

[CT89] M.-H. Chan and H.-T. Tsui, "Recognition of partially occluded 3d objects," *IEE Proceedings*, vol. 136, pp. 124–141, March 1989.

[DBM77] S. A. Dudani, K. J. Breeding, and R. B. McGhee, "Aircraft identification by moment invariants," *IEEE Tranactions on Computers*, vol. 26, pp. 39–46, January 1977.

[DBS77] R. Davis, B. Buchanan, and E. Shortliffe, "Production rules as a representation for a knowledge-based consultation program," *Artificial Intelligence*, vol. 8, pp. 15–45, 1977.

[DG86] S. R. Dubois and F. H. Glanz, "An autoregressive model approach to two-dimensional shape classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 1, pp. 55–66, 1986.

[Dic87] E. D. Dickmanns, "Object recognition and real-time relative state estimation under egomotion," in *NATO Advanced Research Workshop on Real-Time Object Recognition*, pp. 1–15, August 1987.

[Dic88] E. D. Dickmanns, "An integrated approach to feature based dynamic vision," in *Conference on Computer Vision and Pattern Recognition*, pp. 820–825, IEEE, 1988.

[DKZ79] J. D. Desseimoz, M. Kunt, and J. M. Zurcher, "Recognition and handling of overlapping industrial parts," in *Proceeding of the Ninth International Symposium on Industrial Robots*, pp. 357–366, March 1979.

[Dre77] S. Dreyfus, *The Art and Theory of Dynamic Programming*. New York: Academic Press, 1977.

[Dud76] S. A. Dudani, "Distance weighted k-nearest neighbor rule," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 6, pp. 325–327, April 1976.

[Ett88] G. J. Ettinger, "Large hierarchical object recognition using libraries of parameterized model sub-parts," in *Conference on Computer Vision and Pattern Recognition*, pp. 32–41, IEEE, 1988.

[FB81] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, pp. 381–395, June 1981.

[FH83] O. Faugeraus and M. Hebert, "A 3d recognition and positioning algorithm using geometrical matching between surface primitives," in *Proceeding of the 7th International Joint Conference on Artificial Intelligence*, 1983.

[FHK+82] T. J. Fang, Z. H. Huang, L. N. Kanal, B. Lambird, D. Lavine, G. Stockman, and F. L. Xiong, "Three dimensional object recognition using a transformation clustering technique," pp. 678–681, IEEE, 1982.

[Fis83] R. B. Fisher, "Using surfaces and object models to recognize partially obscured objects," in *International Joint Conference on AI*, pp. 989–995, 1983.

[Fre77] H. Freeman, "Shape description via the use of critical points," in *Conference on Pattern Recognition and Image Processing*, pp. 168–174, IEEE, June 1977.

[FT86] O. D. Faugeraus and G. Toscini, "The calibration problem for stereo," in *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pp. 15–20, IEEE, 1986.

[Fu74] K.-S. Fu, *Syntactic Methods in Pattern Recognition*. New York/London: Academic Press, 1974.

[Ger88] G. Gerig, "Linking image-space and accumulator-space: A new approach for object-recognition," in *Proceedings of the International Conference on Computer Vision*, pp. 112–117, IEEE, 1988.

[GL87] R. R. Goldberg and D. G. Lowe, "Verication of 3-d parametric models in 2-d image data," in *Proceedings of the Workshop on Computer Vision*, pp. 255–257, IEEE, 1987.

[GLP84] W. E. L. Grimson and T. Lozano-Perez, "Model-based recognition and localization from sparse range or tactile data," *International Journal of Robotics Research*, vol. 3, pp. 3–35, Fall 1984.

[GLP85] W. E. Grimson and T. Lozano-Perez, "Recognition and localization of overlapping parts from sparse data in two and three dimensions," in *?*, pp. 61–66, IEEE, 1985.

[GLP87] W. E. L. Grimson and T. Lozano-Perez, "Localizing overlapping parts by searching the interpretation tree," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-9, pp. 469–482, July 1987.

[GM88] P. G. Gottschalk and T. N. Mudge, "Efficient encoding of local shape: Features for 3-d object recognition," in *Proceeding of the SPIE: Intelligent Robots and Computer Vision, Seventh in a Series*, pp. 46–56, 1988.

[Goa83] C. Goad, "Special purpose automatic programming for 3d model-based vision," in *Proceedings of the DARPA Image Understanding Workshop*, (Arlington, VA), pp. 94–104, 1983.

[Gra72] G. H. Granlund, "Fourier preprocessing for hand print character recognition," *IEEE Transactions on Computers*, vol. 21, pp. 195–201, February 1972.

[Gre86] K. Grebner, "Model based analysis of industrial scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 28–31, 1986.

[Gri88a] W. E. Grimson, "The combinatorics of object recognition in cluttered environments using constrained search," in *Proceedings of the International Conference on Computer Vision*, pp. 218–227, IEEE, 1988.

[Gri88b] W. E. Grimson, "On the recognition of parameterized 2-d objects," *International Journal of Computer Vision*, vol. 3, pp. 353–372, 1988.

[Gri88c] W. E. L. Grimson, "On the recognition of curved objects," in *IEEE Conference on Robotics and Automation*, pp. 1414–1420, 1988.

[Gri89] W. E. Grimson, "On recognition of curved objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, pp. 632–643, June 1989.

[GTM87] P. G. Gottschalk, J. L. Turney, and T. N. Mudge, "Two-dimensional partially visible object recognition using efficient multidimensional range queries," in *Proceedings of the IEEE Conference on Robotics and Automation*, pp. 1582–1589, 1987.

[GTM89] P. G. Gottschalk, J. L. Turney, and T. Mudge, "Efficient recognition of partially visible objects using a logarithmic complexity matching technique," *International Journal of Robotics Research*, pp. 110–131, December 1989.

[Guz69] A. Guzman, "Decomposition of a visual scene into three-dimensional bodies," in *Automatic Interpretation and Classification of Images* (A. Grasseli, ed.), New York: Academic Press, 1969.

[Hea86] D. J. Healy, "A skeletonizing algorithm with improved isotropy," in *Proceedings of the Conference on Visual Communications and Image Processing*, pp. 138–145, SPIE Vol. 707, 1986.

[HH87] C. Hansen and T. Henderson, "CAGD-based computer vision," in *Proceedings of the Workshop on Computer Vision*, pp. 100–105, IEEE, 1987.

[HH88] C. Hanson and T. Henderson, "Towards the automatic generation of recognition strategies," in *Proceedings of the International Conference on Computer Vision*, pp. 275–279, IEEE, 1988.

[Hil87] D. W. Hillis, "The connection machine," *Scientific American*, vol. 256, pp. 108–115, June 1987.

[HLZ+87] R. M. Haralick, C. N. Lee, X. Zhuang, V. G. Vaidya, and M. B. Kim, "Pose estimation from corresponding point data," in *Proceedings of the Workshop on Computer Vision*, pp. 258–263, IEEE, 1987.

[Hor83] B. K. P. Horn, "Extended gaussian images," AI Memo 740, MIT AI Laboratory, 1983.

[HT85] J. J. Hopfield and D. W. Tank, "?," *Biological Cybernetics*, vol. 52, pp. 141–152, 1985.

[Hu62] M.-K. Hu, "Visual pattern recognition by moment invariants," *IRE Transactions on Information Theory*, vol. 8, pp. 179–187, February 1962.

[HU88] D. P. Huttenlocher and S. Ullman, "Object recognition using alignment," in *International Conference on Computer Vision*, pp. 102–111, IEEE, 1988.

[Huf71] D. A. Huffman, "Impossible objects as nonsense sentences," in *Machine Intelligence 6* (B. Meltzer and D. Mitchie, eds.), Edinburgh: Edinburgh University Press, 1971.

[Hut88] D. P. Huttenlocher, *Three-Dimensional Recognition of Solid Objects from a Two-Dimensional Image*. PhD thesis, MIT, April 1988.

[HW75] H. Hemami and F. C. Weimer, "Identification of three-dimensional objects by sequential image matching," in *Proceedings of the Conference on Computer Graphics*, pp. 273–278, IEEE, 1975.

[Hwa87] V. S. S. Hwang, "Recognition of two dimensional objects using hypothesis integration technique," in *Proceedings of the Workshop on Computer Vision*, pp. 106–111, IEEE, 1987.

[Jac87] D. W. Jacobs, "GROPER: A grouping based recognition system for two dimensional objects," in *Proceedings of the Workshop on Computer Vision*, pp. 164–169, IEEE, 1987.

[JW84] L. Jacobson and H. Wechsler, "A theory for invariant reconition in the frontoparallel plane," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, pp. 325–331, May 1984.

[Kan78] T. Kanade, "A theory of Origami world," Tech. Rep. CMU-CS-78-144, Computer Science Department, Caregie Mellon University, 1978.

[KD87] M. R. Korn and C. R. Dyer, "3-d multiview object representations for model-based object recogniton," *Pattern Recogntion*, vol. 20, pp. 91–103, 1987.

[KJ86] T. F. Knoll and R. C. Jain, "Recognizing partially visible objects using feature indexed hypotheses," *IEEE Journal of Robotics and Automation*, vol. 2, pp. 3–13, March 1986.

[KJ87] T. F. Knoll and R. C. Jain, "Learning to recognize objects using feature indexed hypotheses," in *Proceedings of the International Conference on Computer Vision*, pp. 552–556, IEEE, 1987.

[KK85] M. W. Koch and R. L. Kashyap, "A vision system to identify occluded industrial parts," in *Conference on Pattern Recognition and Image Processing*, pp. 55–60, IEEE, 1985.

[KK87] M. W. Koch and R. L. Kashyap, "Using polygons to recongize and locate partially occluded objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-9, pp. 483–494, July 1987.

[KSSS86] A. Kalvin, E. Shonberg, J. T. Schwartz, and M. Sharir, "Two-dimensional, model-based boundary matching using footprints," *International Journal of Robotics Research*, vol. 5, pp. 38–55, Winter 1986.

[Kuh84] F. P. Kuhl, "Global shape recognition of 3-d objects using a differential library storage," *Computer Vision, Graphics and Image Processing*, vol. 27, pp. 97–114, 1984.

[KvD79] J. J. Koenderink and A. J. van Doorn, "The internal representation of solid shape with respect to vision," *Biological Cybernetics*, vol. 32, pp. 211–216, 1979.

[KWT87] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," in *Proceedings of the First International Conference on Computer Vision*, pp. 259–268, IEEE, 1987.

[LHD88] S. Linnainmaa, D. Harwood, and L. S. Davis, "Pose determination of a three-dimensional object using triangle pairs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, pp. 634–647, September 1988.

[LHF88] Y. Liu, T. X. Huang, and O. D. Faugeras, "Determination of camera location from 2D to 3D line and point correspondences," in *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pp. 82–88, IEEE, 1988.

[Low85] D. G. Lowe, *Perceptual Organization and Visual Recognition.* Kluwer Academic Publishers, 1985.

[Low87a] D. G. Lowe, "Three-dimensional object recognition from single two-dimensional images," *Artificial Intelligence*, vol. 31, pp. 355–395, 1987.

[Low87b] D. G. Lowe, "The viewpoint consistency constraint," *International Journal of Computer Vision*, vol. 1, no. 1, pp. 57–72, 1987.

[LP88] D. Lee and T. Pavlidis, "One-dimensional regularization with discontinuities," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, pp. 822–829, November 1988.

[LSW88a] Y. Lamdan, J. T. Schwartz, and H. J. Wolfson, "Object recognition by affine invariant matching," in *Conference on Computer Vision and Pattern Recognition*, p. 335, IEEE, 1988.

[LSW88b] Y. Lamdan, J. T. Schwartz, and H. J. Wolfson, "On recognition of 3-d objects from 2-d images," in *IEEE Conference on Robotics and Automation*, pp. 1407–1413, 1988.

[Lu88] Y. Lu, *Reasoning About Edges In Scale Space.* PhD thesis, University of Michigan, April 1988.

[LW66] E. L. Lawler and D. W. Wood, "Branch-and-bound methods: A survey," *Operations Research*, vol. 14, pp. 699–719, 1966.

[LW88a] Y. Lamdan and H. Wolfson, "Geometric hashing: A general and efficient model-based recognition scheme," in *Proceedings of the International Conference on Computer Vision*, pp. 238–248, IEEE, 1988.

[LW88b] Y. Lamdan and H. J. Wolfson, "Geometric hashing," Tech. Rep. Robotics Report No. 152, New York University Dept. of Computer Science Robotics Research Laboratory, May 1988.

[MA77] J. W. Mckee and J. K. Aggarwal, "Computer recognition of partial views of curved objects," *IEEE Transaction on Computers*, vol. 26, no. 8, pp. 790–800, 1977.

[Mat76] J. Mattill, "The bin of parts problem and the ice-box puzzle," *Technology Review*, vol. 78, no. 7, pp. 18–19, 1976.

[MC88] D. W. Murray and D. B. Cook, "Using the orientation of fragmentary 3-d edge segments for polyhedral object recognition," *International Journal of Computer Vision*, vol. 2, pp. 153–169, 1988.

[MF75] P. M. Merlin and D. J. Farber, "A parallel mechanism for detecting curver in pictures," *IEEE Transactions on Computers*, vol. 24, pp. 96–98, January 1975.

[MGA88] E. Mjolsness, G. Gindi, and P. Anandan, "Optimization in model matching and perceptual organization: A first look," Tech. Rep. RR-634, Yale Dept. of Computer Science, 1988.

[Nal88] V. Nalwa, "Representing oriented piecewise c_2 surfaces," in *Proceedings of the International Conference on Computer Vision*, pp. 40–51, IEEE, 1988.

[NB80] R. Nevatia and K. R. Babu, "Linear feature extraction and description," *Computer Vision, Graphics and Image Processing*, vol. 13, pp. 257–269, 1980.

[Nil80] N. J. Nilsson, *Principles of Artificial Intelligence.* Norwood, NJ: Tioga, 1980.

[Oht85] Y. Ohta, *Knowledge-Based Interpretation of Outdoor Natural Scenes.* Boston London Melbourne: Pitman Advanced Publishing Program, 1985.

[One66] B. Oneill, *Elementary Differential Geometry.* New York: Academic Press, 1966.

[Pav77] T. Pavlidis, *Structural Pattern Recognition.* Springer, 1977.

[PD87] H. Plantinga and C. R. Dyer, "The Asp: A continuous viewer-centered representation for 3d object recognition," in *Proceedings of the International Conference on Computer Vision*, pp. 626–630, IEEE, 1987.

[Pea84] J. Pearl, *Heuristics: Intelligent Search Strategies fo Computer Problem Solution.* Reading MA: Addison Wesley, 1984.

[Per77] W. A. Perkins, "Model-based vision system for scenes containing multiple parts," in *Fifth International Joint Conference on Artificial Intelligence*, pp. 678–684, 1977.

[Per78] W. A. Perkins, "A model-based vision system for industrial parts," *IEEE Transactions on Computers*, vol. 27, pp. 126–143, February 1978.

[PF77] E. Persoon and K.-S. Fu, "Shape discrimination using fourier descriptors," *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-7, pp. 170–179, March 1977.

[PFTV86] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes: The Art of Scientific Computing.* Cambridge University Press, 1986.

[PILSL88] G. Pasquariello, M. Iannotta, S. Losito, and G. Sylos-Labini, "A system for 3-d workpiece recognition," in *Proceedings of the International Conference on Computer Vision*, pp. 280–284, IEEE, 1988.

[PTK85] T. Poggio, V. Torre, and C. Koch, "Computational vision and regularization theory," *Nature*, vol. 317, pp. 314–319, 1985.

[Ree81] A. P. Reeves, "The general theory of moments for shape analysis and the parallel implementation of moment operations," Tech. Rep. TR-EE 81-37, Purdue University, October 1981.

[RFC88] T. C. Rearick, J. L. Frawley, and P. P. Cortopassi, "Using perceptual grouping to recognize and locate partially occluded objects," in *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pp. 840–846, IEEE, 1988.

[RK76] A. Rosenfeld and A. C. Kak, *Digital Picture Processing*, pp. 109–123. New York/San Francisco/London: Academic Press, 1976.

[Rob64] L. G. Roberts, "Machine perception of three-dimensional solids," in *Optical and Electro-Optical Information Processing*, pp. 159–197, MIT Press, 1964.

[Rog85] D. F. Rogers, *Procedural Elements for Computer Graphics.* New York: McGraw-Hill, 1985.

[RPT85] A. P. Reeves, R. J. Prokop, and R. W. Taylor, "Recognitive aspects of moment invariants," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, (San Francisco, CA), pp. 452–457, June 1985.

[RT89] A. P. Reeves and R. P. Taylor, "Identification of three-dimensional objects using range information," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, pp. 403–410, April 1989.

[Rut89] R. A. Rutenbar, "Simulated annealing algorithms: An overview," *IEEE Circuits and Devices Magzine*, pp. 19–26, January 1989.

[SA86] B. Shararay and D. J. Anderson, "Optimal smoothing of digitized contours," in *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pp. 210–218, IEEE, 1986.

[SB87] J. H. Stewman and K. W. Bowyer, "Aspect graphs for convex planar-face objects," in *Proceedings of the Workshop on Computer Vision*, pp. 123–130, IEEE, 1987.

[SB90] F. Solina and R. Bajcsy, "Recovery of parametric models from range images: The case for superquadrics with global deformations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, pp. 131–147, February 1990.

[Sca85] L. E. Scales, *Introduction to Non-Linear Optimization.* New York: Springer, 1985.

[Sch86] B. G. Schunk, "Gaussian filters and edge detection," Tech. Rep. GMR-5586, General Motors Research Laboratory, October 1986.

[SD71] J. Sklansky and G. A. Davison, "Recognizing three-dimensional objects by their silhouettes," *SPIE Journal*, vol. 10, pp. 10–17, Oct/Nov/Dec 1971.

[SDH84] T. M. Silberberg, L. Davis, and D. Harwood, "An iterative Hough procedure for three-dimensional object recognition," *Pattern Recognition*, vol. 17, no. 6, pp. 621–629, 1984.

[SE85] G. Stockman and J. C. Esteva, "3d object pose from clustering with multiple views," *Pattern Recognition Letters*, vol. 3, pp. 279–286, July 1985.

[SEB88] L. Stark, D. Eggert, and K. Bowyer, "Aspect graphs and nonlinear optimization in 3-d object recognition," in *Proceedings of the International Conference on Computer Vision*, pp. 501–507, IEEE, 1988.

[Seg83] J. Segen, "Locating randomly oriented shapes from partial views," in *Proceedings of the SPIE Cambridge Syposium on Robot Vision and Sensory Controls*, pp. 676–684, SPIE, 1983.

[Sha85] B. Shahraray, *Measurement of Boundary Curvature of Quantized Shapes: A Method Based on Smoothing Spline Approximation.* PhD thesis, University of Michigan, 1985.

[SHD84] T. M. Silberberg, D. Harwood, and L. S. Davis, "Object recognition using oriented model points," in *First IEEE Conference on AI Applications*, pp. 621–629, 1984.

[SKB82] G. Stockman, S. Kopstein, and S. Bennet, "Matching images to models for registration and object detection via clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 4, May 1982.

[Sol87] F. Solina, *Shape Recovery and Segmentation with Deformable Part Models.* PhD thesis, University of Pennsylvania, December 1987.

[STT87] Y. Sato, R. Tamano, and S. Tamura, "Connectionist approach to 3-d object recognition: Matching of parameterized models and mononcular image," in *Proceedings of the Workshop on Computer Vision*, pp. 252–254, IEEE, 1987.

[SU88] D. Shoham and S. Ullman, "Aligning a model using minimal information," in *Proceedings of the International Conference on Computer Vision*, pp. 259–262, IEEE, 1988.

[SW88] M. Seibert and A. M. Waxman, "Spreading activation layers, visual saccades, and invariant representations for neural pattern recognition," Tech. Rep. LSR-TR-5, Laboratory for Sensory Robotics, College of Engineering, Boston University, Boston, Ma 02215, January 1988.

[TC88] C.-H. Teh and R. T. Chin, "On image analysis by the methods of moments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, pp. 496–513, July 1988.

[Tea80] M. R. Teague, "Image analysis via the general theory of moments," *Journal of the Optical Society of America*, vol. 70, pp. 920–930, August 1980.

[TFF88] L. W. Tucker, C. R. Feynman, and D. M. Fritzsche, "Object recognition using the connection machine," in *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pp. 871–878, IEEE, 1988.

[TM87] D. W. Thompson and J. L. Mundy, "Three-dimensional model matching from and unconstrained viewpoint," in *Proceedings of the Conference on Robotics and Automation*, pp. 208–220, IEEE, 1987.

[TMV85] J. L. Turney, T. N. Mudge, and R. A. Volz, "Recognizing partially occluded parts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 7, pp. 410–421, July 1985.

[TMV86] J. L. Turney, T. N. Mudge, and R. A. Volz, "Solving the bin of parts problem," in *Vision '86 Conference Proceedings*, pp. 4:21–4:38, 1986.

[TP88] M. J. Tarr and S. Pinker, "Mental rotation and orientation-dependence in shape recognition, unpublished paper," tech. rep., MIT Department of Brain and Cognitive Science, 1988.

[TR87] R. W. Taylor and A. P. Reeves, "Classification quality assessment for a generalized model-based object identification system," in *Proceedings of the IEEE Conference on Systems Man and Cybernetics*, pp. 412–417, IEEE, 1987.

[Tro80] H. Tropf, "Analysis-by-synthesis search for segmentation applied to workpiece recognition," in *Fifth International Conference on Pattern Recognition*, pp. 241–244, IEEE, 1980.

[Tur74] K. J. Turner, *Computer Perception of Curved Objects Using a Television Camera.* PhD thesis, University of Edinburgh, 1974.

[Tur86] J. L. Turney, *Recognition of Partially Occluded Parts.* PhD thesis, The University of Michigan, August 1986.

[TWK87] D. Terzopolous, A. Witkin, and M. Kass, "Symmetry-seeking models for 3d object recontruction," in *International Conference on Computer Vision*, pp. 269–276, IEEE, 1987.

[TWK36] D. Terzopoulos, A. Witkin, and M. Kass, "Constraints on deformable models: Recovering 3d shape and nonrigid motion," *Artificial Intelligence*, vol. 36, no. 1, pp. 91–123, 36.

[Van87a] P. Van Hove, "Model-based silhouette recognition," in *Proceedings of the Workshop on Computer Vision*, pp. 88–93, IEEE, 1987.

[Van87b] P. Van Hove, "Silhouette slice theorems," Tech. Rep. TR-764, MIT Lincoln Laboratory, Lexington, MA, 1987.

[Wal72] D. I. Waltz, *Generating Semantic Descriptions from Drawings of Scenes with Shadows*. PhD thesis, MIT AI Lab, 1972.

[Wei88] I. Weiss, "Projective invariants of shapes," in *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pp. 291–297, IEEE, 1988.

[Whi88] G. Whitten, "Vertex space and its application to model based object recognition," in *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pp. 847–857, IEEE, 1988.

[Wie83] J. S. Wiejak, "Moment invariants in theory and practice," *Image and Vision Computing*, vol. 1, pp. 79–84, May 1983.

[WM80] T. P. Wallace and O. R. Mitchell, "Analysis of three-dimensional movement using fourier descriptors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-2, pp. 583–588, November 1980.

[WM85] R. J. Watt and M. J. Morgan, "A theory of the primitive spatial code in human vision," *Vision Research*, vol. 25, no. 11, pp. 1661–1674, 1985.

[WMF81] T. P. Wallace, O. R. Mitchell, and K. Fukunaga, "Three-dimensional shape analysis using local shape descriptors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-3, pp. 310–323, May 1981.

[WS82] L. T. Watson and L. G. Shapiro, "Identification of space curves from two-dimensional perspective views," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-4, pp. 469–475, September 1982.

[WW80] T. P. Wallace and P. A. Wintz, "An efficient three-dimensional aircraft recognition algorithm using normalized fourier descriptors," *Computer Vision, Graphics and Image Processing*, vol. 13, pp. 99–126, 1980.

[YMA80] K. R. Yam, W. N. Martin, and J. K. Aggarwal, "Analysis of scenes containing several occluding curvilinear objects," Tech. Rep. TR-135, The University of Texas at Austin Department of Computer Science, Austin, TX 78712, February 1980.