INTERPRETING IMAGES OF POLYHEDRAL OBJECTS IN THE PRESENCE OF UNCERTAINTY

BY

ILAN M. SHIMSHONI

B.S., Hebrew University of Jerusalem, 1984
M.S., Weizmann Institute of Science, 1989

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1995

Urbana, Illinois

ABSTRACT

This thesis addresses various problems in computer vision which are related to the interpretation of images of polyhedral objects with a special emphasis on the effects of uncertainty. Its contributions span three areas: we first present an approach to the recovery of 3D shape from a single image using line-drawing analysis and complex reflectance models. The algorithm deals explicitly with uncertainty in vertex position. We then propose an algorithm for computing the finite-resolution aspect graph of polyhedral objects. For each region of the aspect graph a representative finite-resolution aspect is computed. Neighboring regions with identical finite-resolution aspects are merged producing the finite-resolution aspect graph. Finally, we develop a probabilistic approach to object recognition. Match hypotheses are ranked by the probability that they are correct. For each hypothesis the pose of the object is recovered and the region of the pose space compatible with the image uncertainty is computed. Hypotheses which match different features of the same model reinforce each other when the corresponding uncertainty regions in the pose space have a non-empty intersection. Sets of consistent hypotheses are ranked by probability that they are the correct interpretation of the features, producing an ordering of the possible interpretations. The three algorithms have been fully implemented and examples are presented.

To my wife, Efrat and my chidren, Elee and Tamir.

ACKNOWLEDGMENTS

I would like to thank all the people who made this dissertation possible.

My special thanks go to Professor Jean Ponce, my thesis advisor, for his guidance and friendship. Suggestions he made during our many discussions significantly improved the quality of this dissertation.

I would like to thank the other members of my committee Narendra Ahuja, Seth Hutchinson and Gerald DeJong, for showing interest in my research topic and providing helpful comments.

I would especially like to thank Seth Hutchinson, Jon Gratch and Steve LaValle for probabilistic discussions which improved my understanding of probability and statistics and contributed to this dissertation.

I would also like to thank all my officemates, especially the "old guard" Michael Barbehenn, the Steves (Sullivan and LaValle) and the Castaños (Becky and Andres), who have been with me from the beginning, my removed officemate Tanuja Joshi, and my partners for culinary adventures, Jon Gratch, Dan Oblinger and Melinda Gervasio.

Finally, I would like to thank my wife Efrat and my children Elee and Tamir, who gave me other reasons to live besides working on this dissertation, and my parents and parents in-law for their constant support.

This thesis was supported in part by the Beckman Institute and the Center for Advanced Study of the University of Illinois at Urbana-Champaign, by the National Science

Foundation under grant IRI-9224815, and by the National Aeronautics and Space Administration under grant NAG 1-613.

TABLE OF CONTENTS

APPENDIX **Page**

CHAPTER 1

Introduction

Uncertainty manifests itself in most aspects of computer vision. This is a natural consequence of the fact that image interpretation is based on sensing information: uncertainty may be due to image acquisition (e.g., camera distortion, signal intensity quantization, and errors caused by image quantization into a grid of pixels) or to model incompleteness (e.g., simplified reflectance or camera models). As a result, the output of low-level processes (e.g., edge detection algorithms) may itself be inaccurate or incomplete, and high-level processes which use this output must take the corresponding uncertainties into account. Thus, in order to create more robust computer vision algorithms, the effects of uncertainty have to be understood and modeled. In this thesis we deal with various types of uncertainty which come up in problems dealing with the interpretation of images of polyhedral objects. Specifically we address the following three problems: (1) shape recovery of 3D polyhedra using line-drawing analysis and shading information, (2) aspect graph computation for polyhedral objects under finite-resolution camera models, and (3) 3D object recognition. We attack these problems independently, and the relationships between them are discussed in the conclusion.

The thesis is organized as follows: in Chapter 2, we develop an approach to 3D polyhedra shape recovery using line-drawing analysis and shading information [71, 79]. This

algorithm is an extension of Sugihara's classical method for quantitative line-drawing interpretation [85]. It explicitly deals with uncertainty in vertex position (this was not done in the original algorithm [85]), and applies the shape recovery scheme to complex reflectance models including the Lambertian model with interreflections [31, 63] and a reflectance model for specular objects [10, 62, 88]. The algorithm has been fully implemented. We show through examples how uncertainty in vertex position is dealt with correctly, and demonstrate the algorithm's performance on real images of objects with complex reflectance properties.

In Chapter 3, we compute aspect graphs [54] of polyhedral objects which take into account the image spatial quantization [78]. Equations for finite-resolution visual events are first derived. We then use these equations to trace the visual-event curves [55] and compute the corresponding regions and representative aspects for each region using a plane-sweep algorithm [34, 72]. We compute for each characteristic aspect a finite-resolution version by merging features which are closer than a preset distance. This produces a more realistic aspect graph than the classical one. The details of this algorithm are presented in Appendix A. The complexity of the algorithm and the size of the finite-resolution aspect graph are analyzed. The algorithm has been fully implemented and tested on a number of models.

In Chapter 4, we develop a 3D object recognition algorithm [80]. In order to guide the recognition process we compute the probability that a match hypothesis between image and model features is correct. We use the probabilistic peaking effect of measured angles and ratios of lengths [11, 13, 15] to compute these probabilities and account for various types of uncertainty, e.g., incomplete and inexact edge detection. For each match hypothesis we recover the pose of the object and the pose uncertainty region which is due to the uncertainty in vertex position, then find sets of hypotheses reinforcing each other by matching features of the same object with compatible uncertainty regions. We develop

a probabilistic expression to compare these hypothesis sets, and use that expression to rank them. The hypothesis sets with the highest ranks are returned by the algorithm. The algorithm has been fully implemented, and tested on several real images.

Although we attack three different problems in this thesis, a number of techniques are used throughout, and certain algorithms developed in one chapter are used in another. Details on these interrelations, the significance of this thesis, its main contributions and future research directions are discussed in Chapter 5.

We review in Appendix B homotopy continuation [61] and a technique for curve tracing [55] which are used throughout this thesis.

In the rest of this thesis we will denote vectors using bold face type. For most other variables we will use regular type letters.

CHAPTER 2

Recovering The Shape of Polyhedra Using Line-Drawing Analysis and Complex Reflectance Models

2.1 Introduction

We address the problem of deciding whether a labelled line-drawing [20, 26, 39, 44, 65, 89] represents a picture of some polyhedron, and, in this case, reconstructing a polyhedron that projects onto the line-drawing [23, 45, 49, 50, 58, 85, 86] using shading information. We propose an approach that combines Sugihara's algebraic characterization of line-drawings of polyhedra [77, 85, 86] with gradient space constraints [23, 45, 49, 50, 58], shape-from-shading techniques [31, 63], and radiosity methods [21, 59]. Its novelty is that it explicitly takes into account uncertainty in vertex position and phenomena such as interreflections. Its main advantages are its efficiency in using linear programming to reject incorrect scene interpretations and its ability to perform shape recovery using real images of objects with complex reflectance properties [10, 31, 63, 62, 88].

The rest of the chapter is organized as follows. Section 2.2 discusses the state of the art in line-drawing analysis and 3D shape recovery. We introduce our approach in

Section 2.3. We present an algorithm for line-drawing analysis which takes into account uncertainty in vertex position in Section 2.4, and present our 3D shape recovery algorithm in Section 2.5. Implementation and results are presented in Section 2.6. Finally, a number of issues raised by our algorithm and its implementation are discussed in Section 2.7.

2.2 Literature Review

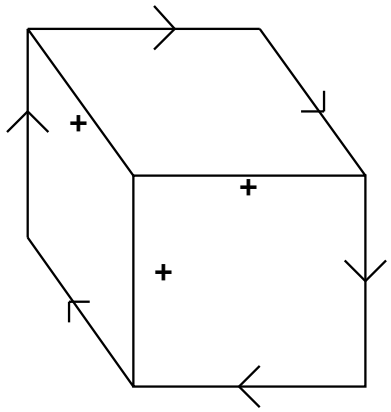


Figure 2.1 Labelling of a line-drawing of a cube.

2.2.1 Line-Drawing Analysis

The problem of labelling a line-drawing and deciding whether a labelled line-drawing represents a correct picture of some polyhedron has been studied extensively [20, 26, 39, 44]. Given a line-drawing of a polyhedron, the labelling problem is to correctly label each edge in the line-drawing as convex (+ label), concave (−), or occluding (← or →). For occluding edges, the occluding face lies on the right when one looks in the direction of the arrow label. For example, Figure 2.1 shows the labelled line-drawing of a cube floating in space. The outer edges are occluding ones, while the inner ones are convex. Several algorithms have been proposed to solve this problem. For example Huffman [44] and Clowes [20] study objects with trihedral junctions (Figure 2.2), and show that only

a small number of the possible labellings for each type of junction actually occurs in polyhedral objects. In [89], Waltz presents a constraint propagation algorithm which exploits these constraints to efficiently find all the consistent labellings of a line-drawing.

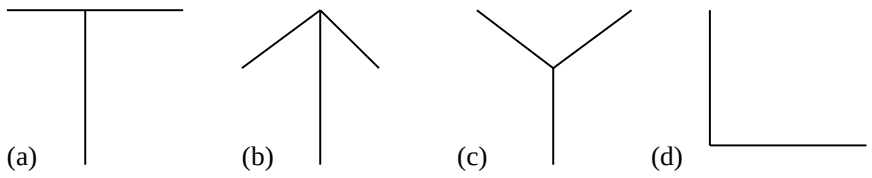


Figure 2.2 Types of trihedral junctions: (a) T junction, (b) arrow junction, (c) fork junction (d) L junction.

The main problem that arises with line-drawing labelling is that even line-drawings that have consistent labellings are not guaranteed to be a picture of a real polyhedron. For example, Figure 2.3(a) shows a line-drawing with a consistent labelling, but admits no correct interpretations with planar faces (Figure 2.3(b)), because any edges between two planar faces must be collinear, which is not the case for faces A and B in the figure. Quantitative approaches using vertex position information are used to address this problem. Each face appearing in the image is parameterized by $z = px + qy + r$, where the pair (p, q) is the gradient of the face, a point in gradient space. The legality of the line-drawing is shown by proving that a set of consistent parameters can be found for all the faces in the image. These methods use equality and inequality constraints on the gradient space, known as gradient space constraints. Mackworth [58], Huffman [45], Kanade [49, 50], and Draper [23] use these constraints to determine whether a line-drawing could be a picture of a polyhedral object.

In [85, 86], Sugihara proved that a labelled line-drawing correctly represents the projection of some polyhedron if and only if the linear constraints

$$\begin{cases} A\mathbf{w} = 0, \\ B\mathbf{w} \geq 0, \end{cases} \tag{2.1}$$

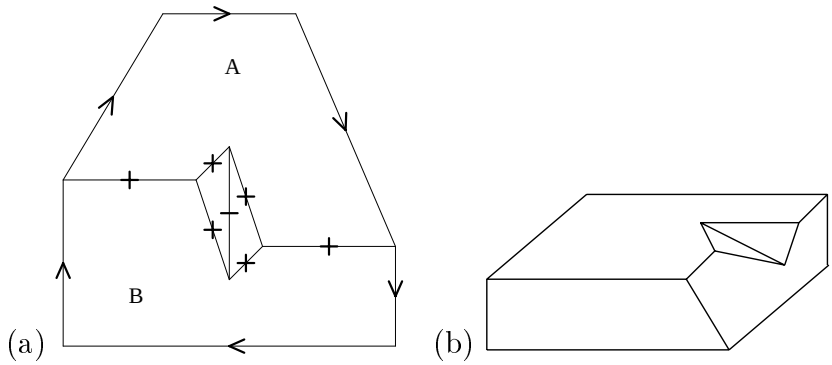


Figure 2.3 A line-drawing with a consistent labelling that only non-polyhedral objects yield: (a) a line-drawing with a consistent labelling; (b) a non-polyhedral object which yields this line-drawing.

admit a solution. These are the *fundamental equations* associated with the line-drawing. The matrices A and B are derived from the positions of the vertices, the incidence relations between vertices, edges, and faces, and the edge labels. The vector $\mathbf{w}$ denotes the unknown face parameters of the observed polygon.

Thus, linear programming can be used to determine whether a line-drawing is "correct", i.e., whether there exists some polyhedron projecting onto it [86]. Due to the loss of depth information in the imaging process, a correct line-drawing admits an infinite number of interpretations. In other words, if there exists a solution $\mathbf{w}$ to (2.1), then there exists an infinite number of distinct solutions. However, when additional cues such as intensity or texture are available, it is possible to select a unique 3D interpretation $\mathbf{w}$ through non-linear optimization under the linear constraints (2.1) [85].

Sugihara's approach is rigorous and elegant. It has been called "the final breakthrough in quantitative (line-drawing) analysis" [43]. However, as remarked by Sugihara himself, a problem with his method is that condition (2.1) is too strict: minute perturbations of vertex positions can make a line-drawing incorrect. For example, the labelled line-drawing in Figure 2.4(a) appears to correctly represent a truncated pyramid seen from above and floating in space. Closer inspection reveals that, due to error in vertex position,

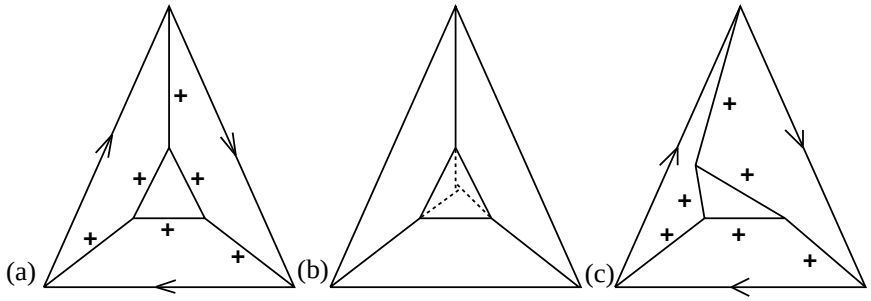


Figure 2.4 (a) A "correct looking" labelled line-drawing. (b) Superstrictness. (c) An incorrect line-drawing.

the three edges that (once extended) should intersect at the pyramid's apex actually do not meet (Figure 2.4(b)).

This "superstrictness" problem is in fact common to most quantitative approaches to line-drawing analysis, including those based on gradient space [23]. To avoid it, Sugihara proposed to detect and delete the constraints that lead to superstrictness by using the purely combinatorial notion of *position-free incidence structures* [85]. This, unfortunately, leads to a new difficulty: The remaining conditions may not be strict enough, and incorrect line-drawings such as the one in Figure 2.4(c) may be classified as correct. In addition, the missing constraints introduce gaps in the recovered surfaces [85].

In this chapter we will avoid superstrictness by explicitly accounting for uncertainty in vertex position. Unlike Sugihara, we do not eliminate constraints that lead to a superstrict set of equations. Instead, we explicitly introduce uncertainty in these constraints. A linear form is obtained by examining the constraints imposed by edges in gradient space [23, 45, 49, 50, 58], yielding a system of equalities and inequalities similar to (2.1). A necessary condition for a line-drawing to be the correct projection of a polyhedron is that this system admits a solution. It can be tested, as before, through linear programming.

2.2.2 Shape From Shading and Reflectance Models

As mentioned earlier, the line-drawing itself does not contain enough information for 3D shape recovery, and for each legal line-drawing there are an infinite number of shapes which could yield that line-drawing (Figure 2.5). Therefore more information such as shading is needed to recover the 3D shape. The problem of recovering shape from shading has been extensively studied [42, 43, 47, 53]. Most approaches assume a Lambertian reflectance which models matte surfaces. According to that model the intensity of light reflected from a surface is proportional to the cosine of the angle between the direction of the light source and the normal to the surface and does not depend on the viewer position. Assuming that the surface of object is Lambertian, the main problem is that the shading information at a point in the image provides only one constraint on surface orientation, while surface orientation has two degrees of freedom. The shape can be recovered by adding the assumption that the object is smooth and therefore the orientation of the surface is continuous. Several methods have been proposed to solve this problem. The traditional method for solving problems mathematically similar to this one depends on growing characteristic strips [43]. In [47], Ikeuchi and Horn present a relaxation method on a grid. In [94], Woodham presents the photometric stereo method which uses several images taken with different lighting directions. The constraints obtained from each image are combined to yield the surface orientation.

The above applications have shown that the Lambertian model does reasonably well in describing pure diffuse reflection. There are many cases however where this model is not adequate. Forsyth and Zisserman [31] study the effects of interreflections between Lambertian surfaces on the intensity of light reflected from them. Nayar et. al. [62] present a shape-from-shading algorithm which specifically deals with interreflections.

One of the important factors which determines the reflectance is the roughness of the surface. Nayar and Oren [64] present a reflectance model for rough matte surfaces. For

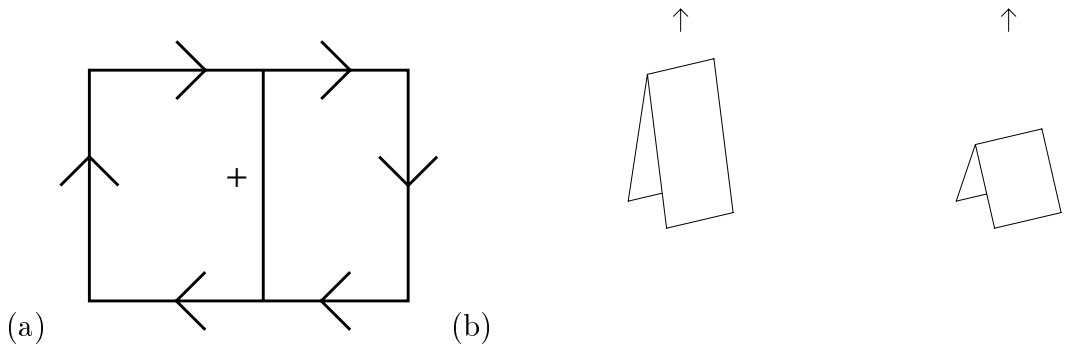


Figure 2.5 An infinite number of shapes which yield the same line-drawing: (a) a line-drawing; (b) two of the infinite number of shapes which yield this line-drawing.

very smooth (mirror-like) specular surfaces, most of the reflected light is concentrated in the specular direction. Beckmann and Spizzichino [10] and Torrance and Sparrow [88] present reflectance models which deal with rough specular surfaces. Nayar et. al. [63] compare the two models and present a general model which simplifies these models and combines them with the Lambertian reflectance model. These reflectance models are used in several graphics and shape-from-shading algorithms. Cook and Torrance [22] present a modified version of the Torrance-Sparrow model for rendering images of objects; Healey and Binford [41] use the Torrance-Sparrow model to determine local shape from specular reflections; and Tagare and deFigueiredo [87] use it to recover the shape and reflectance of surfaces.

In our case we are not able to use the shape-from-shading methods described above because the smoothness assumption is violated at the edges of the object, and the shading over faces of the object is constant which does not enable us to recover the surface orientation. We therefore develop special shape recovery techniques for polyhedral objects which use the line-drawing and the shading information.

2.3 Approach

2.3.1 Notation

We consider a labelled line-drawing, represented by a set of faces, vertices, labelled edges, and their incidence relations. Orthographic projection is assumed. The image is in the x, y plane, and the projection direction is along the z axis.

A face f is represented by its equation:

$$z = px + qy + r, \tag{2.2}$$

and the gradient vector (p, q) is denoted by $\mathbf{g}$.

A vertex v is represented by its image position (x, y) and its (unknown) depth z. In the presence of uncertainty, the actual position (x, y) is not known exactly. Instead, it is related to the measured image position $(\tilde{x}, \tilde{y})$ through a perturbation (μ, ν), with the constraints:

$$\begin{cases} x = \tilde{x} + \mu, & |\mu| \leq \epsilon, \\ y = \tilde{y} + \nu, & |\nu| \leq \epsilon. \end{cases} \tag{2.3}$$

In other words, the actual position is within some small rectangle of side 2ϵ from the measured position.

An edge e is represented by its endpoints v_i, v_j and its adjacent faces f_k, f_l. It is oriented from v_i to v_j, and f_k (resp. f_l) lies on its right (resp. its left).

2.3.2 Sugihara's Fundamental Equations

Assume for the moment that the image coordinates of the vertices are known perfectly. Consider the line-drawing in Figure 2.6(a). Faces f_1 and f_2 meet along concave edge e_1. Vertex v_1 lies on e_1, and thus on f_1, f_2, while v_2 lies on f_2 only. Let the equation of f_i be

$z = p_i x + q_i y + r_i$ and the coordinates of v_j be (x_j, y_j, z_j), we have the following equality constraints:

$$\begin{cases} z_1 = p_1 x_1 + q_1 y_1 + r_1, \\ z_1 = p_2 x_1 + q_2 y_1 + r_2, \\ z_2 = p_2 x_2 + q_2 y_2 + r_2. \end{cases} \tag{2.4}$$

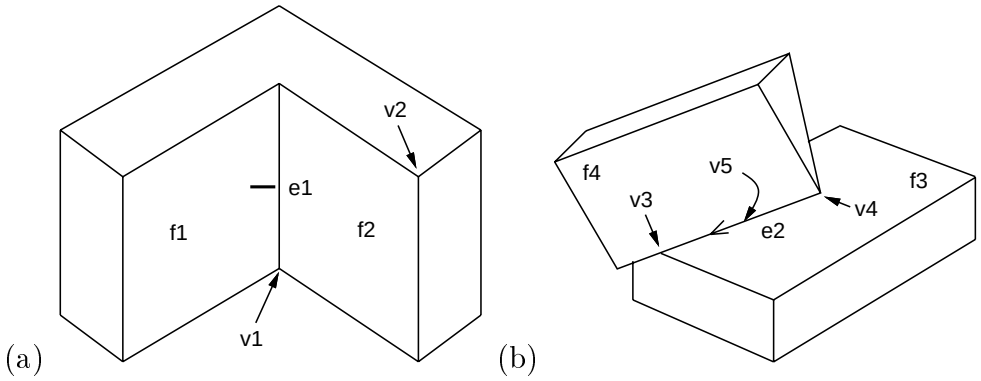


Figure 2.6 Line-drawings and constraints.

Since e_1 is concave, v_2 must lie *above* the plane of f_1, yielding the inequality constraint:

$$z_2 > p_1 x_2 + q_1 y_2 + r_1. \tag{2.5}$$

Consider now the line-drawing in Figure 2.6(b). Face f_4 occludes f_3 along e_2. Let v_3 and v_4 be two consecutive vertices along e_2, and v_5 be their mid-point, we have the following inequalities:

$$\begin{cases} z_3 \geq p_3 x_3 + q_3 y_3 + r_3, \\ z_4 \geq p_3 x_4 + q_3 y_4 + r_3, \\ z_5 > p_3 x_5 + q_3 y_5 + r_3. \end{cases} \tag{2.6}$$

Equality is allowed in the first two inequalities, but not in the third one. This is because the occluding edge e_2 can touch f_3 at some point, but not at every point (otherwise it would be a concave edge) [85]. Three equality constraints should be added, corresponding to the fact that v_3, v_4, v_5 belong to f_4.

The above constraints are linear in the unknowns p_i, q_i, r_i, z_j. Sugihara gathered all such constraints into (2.1), where A and B are matrices whose coefficients are functions of the vertex coordinates x_j, y_j, and $\mathbf{w}$ is obtained by concatenating the unknowns for all faces and vertices. He proved that a necessary and sufficient condition for a line-drawing to correctly represent the projection of a polyhedron is that these equations admit a solution.

In the next section, we will derive similar constraints in the case where the x, y coordinates of the vertices are not known exactly.

2.4 Constraints under Uncertainty

We first examine the constraints imposed by incident vertices and faces, and then derive gradient space constraints associated with convex and concave edges.

2.4.1 Vertex Constraints

Writing that a vertex v_i lies on two faces f_k and f_l and eliminating the unknown depth z_i yields:

$$(p_k - p_l)x_i + (q_k - q_l)y_i + (r_k - r_l) = 0, \tag{2.7}$$

which can be rewritten as:

$$(p_k - p_l)\tilde{x}_i + (q_k - q_l)\tilde{y}_i + (r_k - r_l) + (p_k - p_l)\mu_i + (q_k - q_l)\nu_i = 0. \tag{2.8}$$

The quadratic terms can be eliminated by introducing the new variables $a_{ki} = p_k\mu_i, b_{ki} = q_k\nu_i$:

$$(p_k - p_l)\tilde{x}_i + (q_k - q_l)\tilde{y}_i + (r_k - r_l) + (a_{ki} - a_{li}) + (b_{ki} - b_{li}) = 0. \tag{2.9}$$

By (2.3) a_{ki}, b_{ki} satisfy non-linear constraints:

$$\begin{cases} |a_{ki}| \leq \epsilon |p_k|, \\ |b_{ki}| \leq \epsilon |q_k|. \end{cases} \tag{2.10}$$

A similar derivation gives the following constraint for occluding edges:

$$(p_k - p_l)\tilde{x}_i + (q_k - q_l)\tilde{y}_i + (r_k - r_l) + (a_{ki} - a_{li}) + (b_{ki} - b_{li}) > 0. \tag{2.11}$$

The next subsection shows how to use gradient space constraints to eliminate the four quadratic constraints.

2.4.2 Edge Constraints

Consider an edge e with extremities v_i, v_j, and two adjacent faces f_k, f_l. Assume that e is either convex or concave. Writing that v_i and v_j belong to f_k and f_l, and eliminating the unknown depths z_i, z_j yields:

$$(\mathbf{g}_k - \mathbf{g}_l) \cdot \mathbf{t}_e = 0, \tag{2.12}$$

where $\mathbf{t}_e$ is the vector joining the projections of the vertices v_i and v_j into the image plane. This is the well known constraint imposed by an edge on the gradients of the adjacent faces [50].

Observe that $\mathbf{t}_e$ is necessarily contained in the double cone joining the uncertainty regions centered in $\tilde{\mathbf{x}}_i = (\tilde{x}_i, \tilde{y}_i)$ and $\tilde{\mathbf{x}}_j = (\tilde{x}_j, \tilde{y}_j)$ (Figure 2.7).

Let $\tilde{\mathbf{t}}_e = \tilde{\mathbf{x}}_j - \tilde{\mathbf{x}}_i$ be the axis of this cone, α be its half-angle, and let $\tilde{\mathbf{n}}_e$ be the vector obtained by rotating $\tilde{\mathbf{t}}_e$ 90° counterclockwise. Since $\mathbf{g}_k - \mathbf{g}_l$ and $\mathbf{t}_e$ are orthogonal, it follows that $\mathbf{g}_k - \mathbf{g}_l$ lies in the double-cone centered on $\tilde{\mathbf{n}}_e$ with half-angle α, i.e.,

$$|(\mathbf{g}_k - \mathbf{g}_l) \cdot \tilde{\mathbf{t}}_e| \leq |(\mathbf{g}_k - \mathbf{g}_l) \cdot \tilde{\mathbf{n}}_e| \tan \alpha. \tag{2.13}$$

This condition is not linear. Next, we show that the sign of $(\mathbf{g}_k - \mathbf{g}_l) \cdot \tilde{\mathbf{n}}_e$ is actually known, which brings us back to the realm of linear algebra.

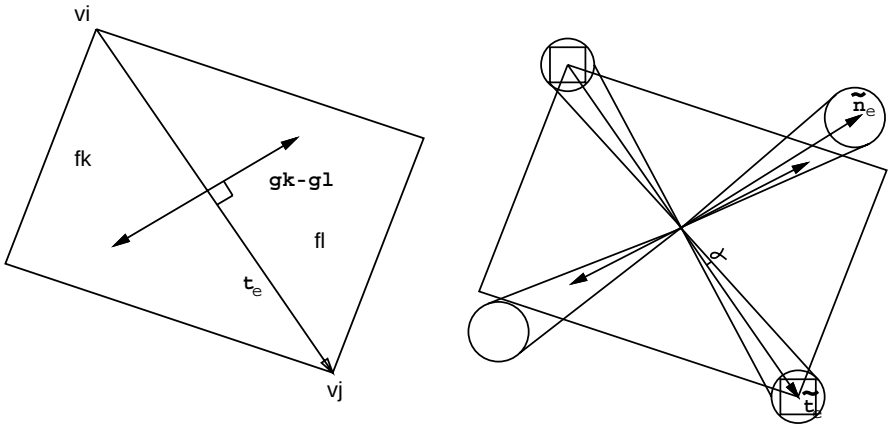


Figure 2.7 Edge constraints. The double arrow indicates that the orientation of $\mathbf{g}_k - \mathbf{g}_l$ is unknown.

We have not used the convexity (or concavity) of the edge yet. Suppose e is convex, let $\mathbf{n}_e$ be the real direction of its normal, and z_l and z_k be the points where the line parallel to the z axis in $\mathbf{x}_i + \mathbf{n}_e$ intersects the planes of f_l and f_k (Figure 2.8).

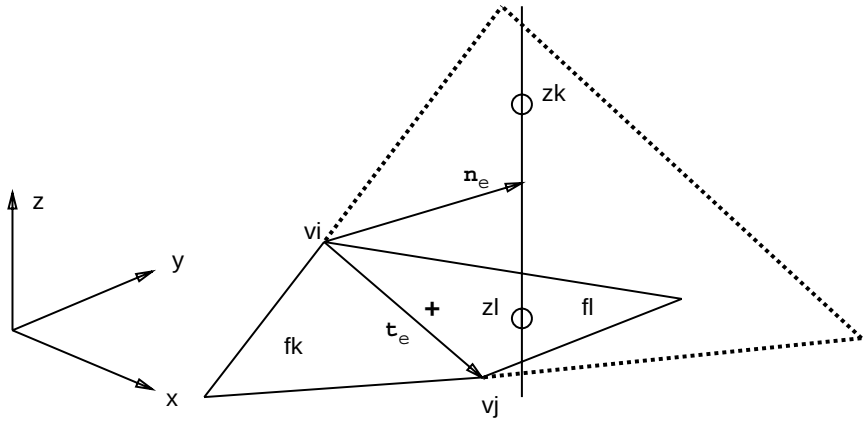


Figure 2.8 The geometric constraints imposed by a convex edge.

Since e is convex, we know that $z_k > z_l$, yielding, after some algebraic manipulation:

$$(\mathbf{g}_k - \mathbf{g}_l) \cdot \mathbf{n}_e > 0, \tag{2.14}$$

which determines the side of the double-cone where $\mathbf{g}_k - \mathbf{g}_l$ lies (Figure 2.9). This is again a well known constraint [50]. We already know that $\mathbf{g}_k - \mathbf{g}_l$ and $\mathbf{n}_e$ are aligned, so:

$$(\mathbf{g}_k - \mathbf{g}_l) \cdot \tilde{\mathbf{n}}_e > 0, \tag{2.15}$$

except in the degenerate case where the two vertices are so close to each other as to make $\alpha = \pi/2$.

Combining (2.13) and (2.15) yields a *linear* constraint:

$$|(\mathbf{g}_k - \mathbf{g}_l) \cdot \tilde{\mathbf{t}}_e| \leq ((\mathbf{g}_k - \mathbf{g}_l) \cdot \tilde{\mathbf{n}}_e) \tan \alpha. \tag{2.16}$$

The analysis applies to concave edges by inverting the orientation of $\mathbf{g}_k - \mathbf{g}_l$ (Figure 2.9).

When the double cone centered in $\tilde{\mathbf{n}}_e$ does not contain the x axis, (2.16) determines the sign $s = \mp 1$ of $p_k - p_l$, yielding the *linear* constraint:

$$|a_{ki} - a_{li}| \leq \epsilon s(p_k - p_l). \tag{2.17}$$

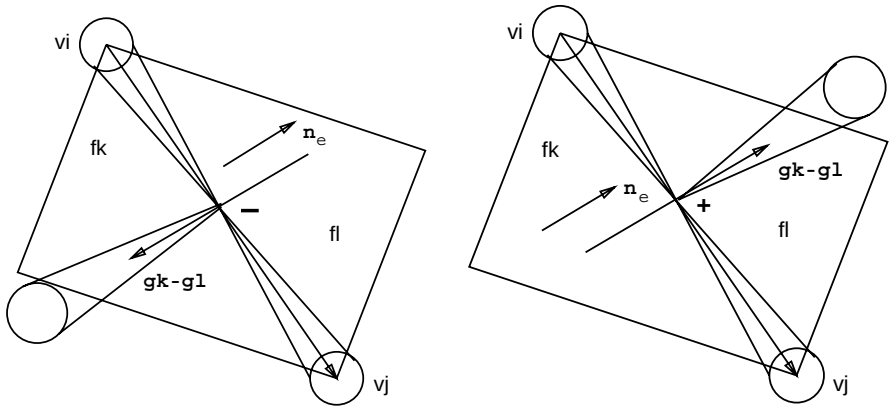


Figure 2.9 Constraints on the direction of $\mathbf{g}_k - \mathbf{g}_l$ imposed by concave (left) and convex (right) edges.

Similarly, when the double cone does not contain the y axis, (2.16) determines the sign s of $q_k - q_l$, yielding:

$$|b_{ki} - b_{li}| \leq \epsilon s(q_k - q_l). \tag{2.18}$$

In general, a convex or concave edge provides at least one of these two constraints for each of its extremities.

2.4.3 Remarks

We have shown that, even in the presence of uncertainty, a labelled line-drawing can be characterized by a set of linear constraints. Let us close this section with a few remarks. First, one of the variables r_k can be set to some arbitrary value since absolute depth cannot be recovered under orthographic projection. The variables z_i can be recovered from the other variables. Second, (2.11) and (2.15) contain strict inequalities, which is normally a difficulty for linear programming. As remarked by Sugihara [86], all the constraints are homogeneous, and strict inequalities with a zero constant term can be replaced by non-strict inequalities with an arbitrary positive constant term. Third, these constraints only provide a necessary condition for correctness. In particular, the signs of $p_k - p_l$ and $q_k - q_l$ cannot always be determined.

2.5 3D Shape Recovery

2.5.1 Principle

Like Sugihara [85], we formulate 3D shape recovery as an optimization problem. Consider a set of visual cues, such as intensity or texture. Let $\mathbf{w}$ denote the unknown geometric parameters, $\mathbf{l}$ denote the other scene parameters (e.g., light source direction and intensity, surface albedo), I_k denote the observed value of the k^{th} cue (e.g., the intensity of face f_k), and $J_k(\mathbf{w}, \mathbf{l})$ denote the value of this cue that would be observed if the actual scene parameters were $\mathbf{w}$ and $\mathbf{l}$. Recovering $\mathbf{w}$ and $\mathbf{l}$ amounts to minimizing:

$$\sum_k (I_k - J_k(\mathbf{w}, \mathbf{l}))^2 \tag{2.19}$$

under the linear constraints derived in Sect. 2.4:

$$\begin{cases} A\mathbf{w} = 0, \\ B\mathbf{w} \geq 0. \end{cases} \tag{2.20}$$

This is what Sugihara calls a quadratic error minimization, a term we find a bit misleading since the error term is in general *not* quadratic in the unknown parameters. This is unfortunate since quadratic minimization under linear constraints can be solved exactly.

To solve the constrained minimization problem, we first use Gaussian elimination to delete all linear equalities and determine the values of a corresponding number of variables as a linear function of the remaining ones. Among the possible choices, we choose the subset of variables which yields the best numerical stability. This is done by finding which n-tuple of standard basis vectors produce a square matrix with the highest possible condition number when added to matrix A.

We solve our constrained optimization problem using an active set method [57], which reduces the original problem to the *unconstrained* minimization of:

$$\sum_k (I_k - J_k(\mathbf{w}, \mathbf{l}))^2 + \sum_j (\min\{0, B_j \mathbf{w}\})^2, \tag{2.21}$$

where B_j denotes the j^{th} row of B. From the way the problem is stated, it is clear that any model J_k which can recover I_k accurately when given the scene parameters $\mathbf{w}$ and $\mathbf{l}$ can be used to recover the shape information and the scene parameters. Therefore models like the ones used in computer graphics to render physically accurate images can be substituted for J_k in the formalization of the problem.

Because optimization algorithms may fall into local minima a method must be devised to give the algorithm a starting position as close as possible to the optimal one. The method we have chosen is to divide the algorithm into two stages. In the first one we run the optimization algorithm using a simplified model (e.g., disregarding the effects of interreflections on the image). Such a model will have fewer unknown parameters than the full model or make it easier to compute $J_k(\mathbf{w}, \mathbf{l})$, but it may give less accurate results. We run the optimization using the simple model with various starting positions. In the second stage we use the best result of the first stage as the starting position for

optimization using the elaborate model. For complex reflectance models we generalized this method for more than two stages, where the result of each stage is the starting position of the next stage.

In order to demonstrate the applicability of our method we have implemented shape recovery from shading information using three reflectance models: the pure Lambertian model, the Lambertian model with interreflections [31, 63], and a reflectance model for smooth specular objects [10, 63, 88].

2.5.2 Lambertian Model

Under the Lambertian model we assume that there is a distant point light source whose direction $\mathbf{s}$ and intensity s are unknown and suppose that the observed object has a constant (but unknown) albedo ρ (see Figure 2.10). Our method differs from classical shape-from-shading techniques [47] in which these parameters are assumed to be known (on the other hand those methods do not assume that the object is polyhedral). Photometric stereo [94] does not require the albedo to be constant or that the object be polyhedral but requires the direction and intensity of the light sources and more than one image.

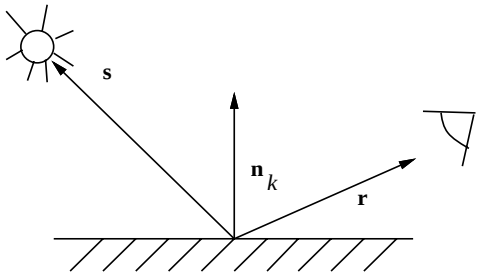


Figure 2.10 The Lambertian reflectance model; $\mathbf{n}_k$ is the normal to the surface, $\mathbf{s}$ is the light source direction, and $\mathbf{r}$ is the viewing direction.

The k^{th} cue I_k is the intensity measured on the k^{th} face. We compute J_k as:

$$J_k(\mathbf{w}, \mathbf{l}) = \frac{s\rho}{\pi}\mathbf{n}_k \cdot \mathbf{s} \tag{2.22}$$

where $\mathbf{n}_k = \frac{1}{\sqrt{(p_k^2+q_k^2+1)}}(-p_k, -q_k, 1)^T$ is the normal to face number k, and the scene parameters $\mathbf{l}$ denote the lighting direction $\mathbf{s}$ the intensity of the light s, and the albedo ρ. In this case we cannot decouple the albedo ρ from the intensity s of the light source, therefore we recover their product.

For this reflectance model we have two stages of optimization, using the result of the first stage as the starting position of the second stage. The simplified model we use in the first stage disregards the uncertainty in vertex position. This reduces the number of unknowns considerably. In the case of superstrictness, like Sugihara [85], we delete the corresponding constraints. In the second stage we take into account all the constraints and the uncertainty in vertex position.

2.5.3 Lambertian Model With Interreflections

Radiosity methods are used in computer graphics to compute the interreflections of visible light and to render physically-accurate images of objects [21, 81]. They have also been used in computer vision to recover the shape of objects from images with interreflections [31, 63]. In [63] photometric stereo is used in the initial stages of the algorithm requiring several images and knowing the light source parameters. On the other hand the albedo is unknown and does not have to be constant, and the shape of the recovered object does not have to be polyhedral.

Under the interreflection model the total irradiance $E(\mathbf{x})$ at point $\mathbf{x}$ is expressed as the sum of the irradiance $E_s(\mathbf{x})$ due to the light source and the irradiance due to all other points on the surface which can see $\mathbf{x}$. The radiance L at the point $\mathbf{x}$ is related to its irradiance E by:

$$L(\mathbf{x}) = \frac{\rho(\mathbf{x})}{\pi} E(\mathbf{x}) \tag{2.23}$$

where $\rho(\mathbf{x})/\pi$ is the bi-directional reflectance distribution function for a Lambertian surface. The geometric factor (the form factor) which relates the irradiance $E(\mathbf{x})$ due to

the radiance of the point $\mathbf{x}'$ to the radiance of $\mathbf{x}'$, whose geometry is illustrated in Figure 2.11 is:

$$F(\mathbf{x}, \mathbf{x}') = \text{View}(\mathbf{x}, \mathbf{x}') \frac{[\mathbf{n} \cdot (\mathbf{x}' - \mathbf{x})][\mathbf{n}' \cdot (\mathbf{x} - \mathbf{x}')]}{|\mathbf{x}' - \mathbf{x}|^4}. \tag{2.24}$$

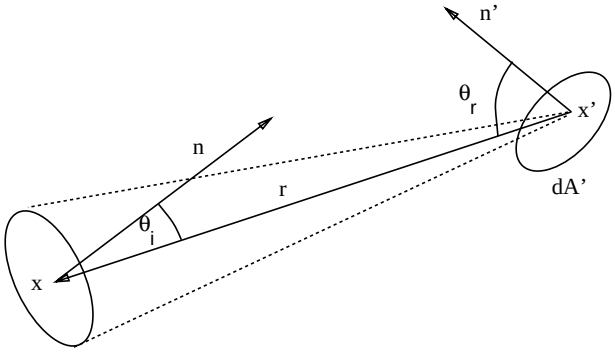


Figure 2.11 The contribution of light reflected from a surface patch dA' around $\mathbf{x}'$ to the irradiance of point $\mathbf{x}$, where $\mathbf{r} = \mathbf{x} - \mathbf{x}'$, and $\mathbf{n}$ and $\mathbf{n}'$ are the surface normals at $\mathbf{x}$ and $\mathbf{x}'$ respectively.

Here, $\text{View}(\mathbf{x}, \mathbf{x}')$ equals one when $\mathbf{x}$ and $\mathbf{x}'$ can see each other, and therefore can illuminate each other, and zero otherwise. Thus the irradiance $E(\mathbf{x})$ of the surface element dA' due to the radiance of the point $\mathbf{x}'$ is:

$$E(\mathbf{x}) = F(\mathbf{x}, \mathbf{x}') L(\mathbf{x}') dA'. \tag{2.25}$$

From equations (2.23) and (2.25), we obtain:

$$L(\mathbf{x}) = L_s(\mathbf{x}) + \frac{\rho}{\pi} \int F(\mathbf{x}, \mathbf{x}') L(\mathbf{x}') dA'. \tag{2.26}$$

We use (2.26) to compute the intensity of the light reflected by the pixels in the image. Each pixel which is a rectangle in the image whose area is D is actually a parallelogram in space whose area is $D/(\mathbf{v} \cdot \mathbf{n})$ where $\mathbf{v}$ is the viewing direction. We assume that each pixel has uniform intensity. Applying (2.26) at a pixel P we obtain:

$$L(P) = s \frac{\rho}{\pi} \mathbf{n}_P \cdot \mathbf{s} + \frac{\rho[\mathbf{v} \cdot \mathbf{n}_P]}{D\pi} \int_{\mathbf{x} \in P} \sum_{P' \neq P} L(P') \int_{\mathbf{x}' \in P'} F(\mathbf{x}, \mathbf{x}') dA' dA \tag{2.27}$$

where $L(P)$ is the intensity of light radiated from pixel P and $\mathbf{n}_P$ is the normal to the surface at P.

When the distance between P and the pixels it can "see" is large compared with the size of P, (2.27) can be approximated by:

$$L(P) = s\frac{\rho}{\pi}\mathbf{n}_P \cdot \mathbf{s} + \frac{\rho}{\pi}\sum_{P' \neq P} \frac{F(\mathbf{x}_P, \mathbf{x}_{P'})D\ L(P')}{\mathbf{v} \cdot \mathbf{n}_{P'}} \tag{2.28}$$

where $\mathbf{x}_P$ is the center of pixel P. For the contribution of each pixel to $L(P)$ we decide which of (2.27) or (2.28) to use by the distance between the pixels. Therefore (2.28) is mainly used for pairs of pixels which lie on both sides of a concave edge, close to it.

For this reflectance model we have three stages of optimization, using the result of each stage as the starting position of the next stage. In the first stage of the optimization we disregard interreflections and the uncertainty in vertex position. For each face f_k we choose the intensity I_k as the minimum intensity value of a pixel of that face. As interreflections can only add to the intensity of a pixel the minimum value should be as close as possible to the light reflected only due to the light source. In the second stage we take into account interreflections. We finally perform a third optimization stage which takes into account vertex uncertainty.

2.5.4 Reflectance Model for Specular Objects

The Lambertian model described above is limited to objects with matte surfaces. To deal with other types of materials, more complete reflectance models have been derived. Beckmann and Spizzichino [10] developed a reflectance model for rough specular objects. In this model the specular reflectance of an object was divided into two components: the specular lobe, and the specular spike. Torrance and Sparrow [88] developed a simpler model which captures only the specular lobe component (this model is widely used in graphics [22] and has also been used computer vision [41, 87]). The relative strengths of

the specular lobe and the specular spike vary with the roughness of the object. As the object is rougher the magnitude of the specular lobe gets stronger and the magnitude of the specular spike decreases. In most instances one of the two specular components is significant while the other is negligible. In addition to these two components, the reflectance of most materials has a diffuse lobe which can be approximated by the Lambertian model.

In [63], Nayar et. al. analyzed the above reflectance models and proposed a simpler model to approximate them, and make them easier to use in computer vision algorithms. According to this model, the irradiance in the $\mathbf{r}$ direction E_{im} is written as:

$$E_{im} = K_{dl}(\mathbf{n} \cdot \mathbf{s}) + \frac{K_{sl}}{\mathbf{n} \cdot \mathbf{r}} \exp(-\frac{\alpha^2}{2\sigma^2}) + K_{ss}\delta(\phi),$$

where (as shown in Figure 2.12) ϕ denotes the angle between $\mathbf{r}$ and the mirror direction s' for source direction $\mathbf{s}$ reflected off the surface whose normal is $\mathbf{n}$, α is the angle between the bisector of ($\mathbf{s}$ and $\mathbf{r}$) and $\mathbf{n}$, σ denotes the slope roughness of the surface, and K_{dl}, K_{sl} and K_{ss} denote the strengths of the diffuse lobe, specular lobe, and specular spike components respectively. The specular spike and the specular lobe components peak when $\mathbf{r} = \mathbf{s}'$ as α and ϕ vanish. As the viewing direction changes the specular spike component decays rapidly to zero. The decay of the specular lobe component is more gradual and depends on the roughness of the surface. The rougher the surface, the larger σ gets, and the slower the decay. In Figure 2.13, polar plots of the three reflection components are shown as a function of the sensor direction for a fixed light direction (Figure 2.13(a)) and as a function of the light source direction for a fixed sensor direction (Figure 2.13(b)).

We have concentrated on relatively smooth objects. In this case most of the specular component of the reflected light is concentrated in directions close to the mirror direction. Faces which have a specular component in their reflection in the direction of the camera are easily distinguished from the others because of the high intensity of light reflected from them. For faces without a specular component the reflectance model reduces to the Lambertian model. For faces with a specular component we know that the camera

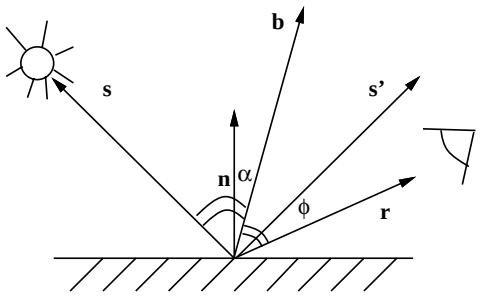


Figure 2.12 The general reflectance model; **n** is the normal to the surface, **s** is the light source direction, **s**′ is the mirror direction, **r** is the viewing direction and **b** is the bisector between **s** and **r**; α is the angle between **n** and **b**, and ϕ is the angle between **r** and **s**′.

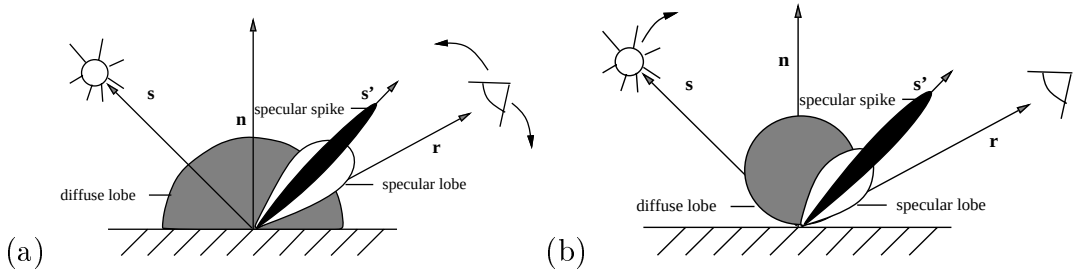


Figure 2.13 Polar plots of the three reflection components (adapted from [63]): (a) a function of the sensor direction for a fixed light source direction; (b) a function of the light source direction for a fixed sensor direction.

is approximately in the mirror direction of that face. Thus given an estimate for the geometric parameters of a specular face, the light source direction **s** can be computed directly. Therefore when a specular face appears in the image, the number of independent unknowns is reduced. The number of unknowns is further reduced when there are several specular faces because that implies that those faces are parallel to each other. So although this reflectance model is more complicated then the Lambertian one, recovering the shape when such faces appear in the image is easier. When there is no such face the algorithm is reduced to the algorithm described for the Lambertian model. For this reflectance model we use the same two stages of optimization as for the Lambertian model.

2.6 Implementation and Results

We have implemented the approach proposed in Sect. 2.4 to test the correctness of a line-drawing in the presence of uncertainty using linear programming. In this case, linear programming is not used for optimization, but rather for deciding whether there exists a point satisfying these constraints. We present results classifying line-drawings as legal or illegal for different uncertainty bounds. We also present our implementation of 3D shape recovery from intensity data using the method described in Sect. 2.5. We have applied an active set optimization method to the solution space obtained through performing Gaussian elimination on the linear equality constraints. However, no special techniques were used exploiting the linearity of the inequalities. This did not degrade the performance of the algorithm because usually a point was found in the early stages of the algorithm that satisfied those constraints, and the main effort was put into computing and minimizing the objective function for points that satisfied all of the inequalities. The algorithm has been implemented in C using the simplex algorithm from [73] and the Levenberg-Marquardt procedure of the MINPACK library [60]. Experiments were run on a SUN SPARC Station.

2.6.1 Line-Drawing Analysis

Figure 2.14(a) shows a sample line-drawing, reproduced from [85]. Of particular interest are the three edges e_4, e_5, e_6 converging toward the center of this truncated pyramid. With the nominal values given in [85], and an $\epsilon = 0$ uncertainty bound, this "correct looking" line-drawing is classified as incorrect by our program (Figure 2.14(b)). The corresponding linear programming problem has 51 variables and 84 constraints. The running time of the program is 3.6s. Next, we set the uncertainty bound to $\epsilon = 0.002$. This time, the line-drawing is classified as correct by our program, as confirmed by Figure

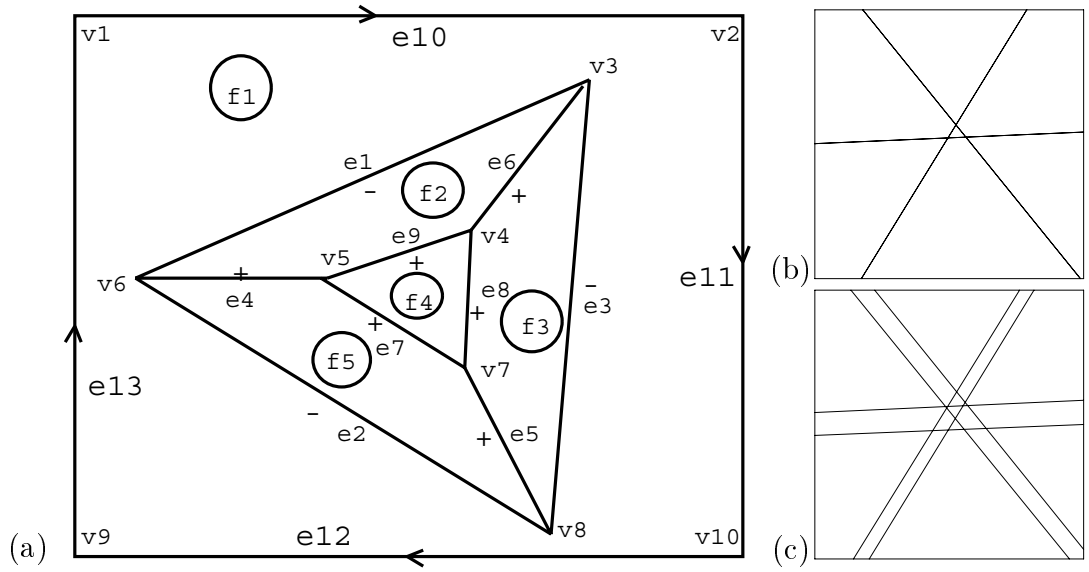


Figure 2.14 (a) A line-drawing from [85]. Close-ups of the region where the support lines of e_4, e_5, e_6 should intersect. (b) Incorrect line-drawing with $\epsilon = 0$. (c) Correct line-drawing with $\epsilon = 0.002$.

2.14(c), where the double cones corresponding to the three edges have been drawn and are shown to intersect. The number of variables is the same as before, but there are 120 constraints, and the running time of the program is 8.2s. The difference in the number of constraints comes from the fact that, for $\epsilon = 0$, each pair of inequalities corresponding to (2.16-2.18) can be replaced by a single equality.

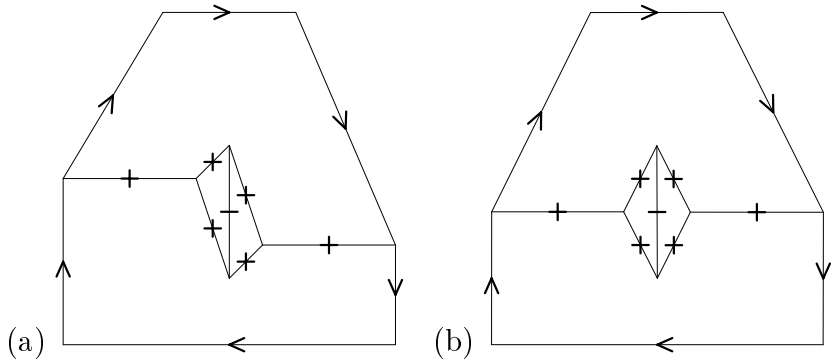


Figure 2.15 Results for two combinatorially equivalent line-drawings: (a) an incorrect line-drawing; (b) a correct line-drawing.

Figure 2.15(a) is reproduced from [9]. It is an incorrect line-drawing, as verified by our program. In this case, the corresponding linear programming problem has 44 variables and 80 constraints, with a total running time of 4.0s. Figure 2.15(b) shows a correct version of the same line-drawing, with a random perturbation of range 0.01 in vertex position. Note that this perturbation is not visible in this picture where the longest edge has length 10. The line-drawing is correctly classified by our program with $\epsilon = 0.01$. The number of variables and constraints is the same as before, and the running time is 3.9s.

2.6.2 Lambertian Objects

We have tested our shape recovery algorithm on several real images. Figure 2.16(a) shows the image of an object made of flat white paper. Edge detection was run on the image and a "clean" line-drawing shown in Figure 2.16(b) was extracted by hand from the results. Figure 2.16(c) shows the reconstruction obtained using the first stage of the algorithm. Notice that some adjacent faces do not coincide. This is due to removal of constraints due to superstrictness. Figure 2.16(d) shows the final result when all constraints have been taken into account and uncertainty in vertex position is reduced to below ϵ. The arrows in Figures 2.16(c,d) show the direction from which the image was taken. In order to obtain a quantitative evaluation of our results we compared the angles between adjacent faces in the original object (which we measured by hand) and in the recovered object. Figure 2.16(e) shows the distribution of the errors by angles.

2.6.3 Lambertian Objects with Interreflections

We have also run the algorithm on a number of objects which have concave edges causing interreflections between adjacent faces. To recover the shape of an object we take a subset of the pixels by uniformly sampling the image. In deciding on the size of the subset there is a tradeoff between the cost of running the algorithm which is proportional to the size

of the subset, and the accuracy of the shape recovery which improves as the size of the selected subset grows. When computing $J_k(\mathbf{w}, \mathbf{l})$ for a certain pixel P, we assume that the intensity $L(P')$ of every other pixel P' in the image is as was measured in the image.

Figure 2.17(a) shows an image of an L-shaped wooden object painted with flat white paint. Figure 2.17(b) shows the intensity reconstruction of the image using the shape and scene parameters recovered by the optimization algorithm. Figure 2.17(c) shows the iso-intensity lines in the original image, and Figure 2.17(d) shows those lines in the rendered image. In Figure 2.17(e) the intensity of a cross-section of the two images is plotted, and Figure 2.17(f) shows two views of the recovered object. The arrows in Figure 2.17(f) show the direction from which the image was taken.

We have also run the algorithm on two images of a styrofoam object (once again painted with flat white paint) which is used for packing computers. The results for the first image (Figure 2.18) are not quite as good as the results for the second image (Figure 2.19), even though for both images the results of the intensity reconstruction are good despite much texture due to the grainy appearance of styrofoam.

In order to obtain a quantitative evaluation of our results we compared the angles between adjacent faces in the original objects (which we measured by hand) and in the recovered objects. For the object in Figure 2.17 the maximum error is 11°. For the object in Figure 2.19 the error is between 12° and 31°. For the object in Figure 2.18 the recovered object is not as close to the original object, and the maximum error is 51° even though the intensity reconstruction is good (Figure 2.18(e)). The distribution of the errors for these three objects is shown in the histograms in Figure 2.20.

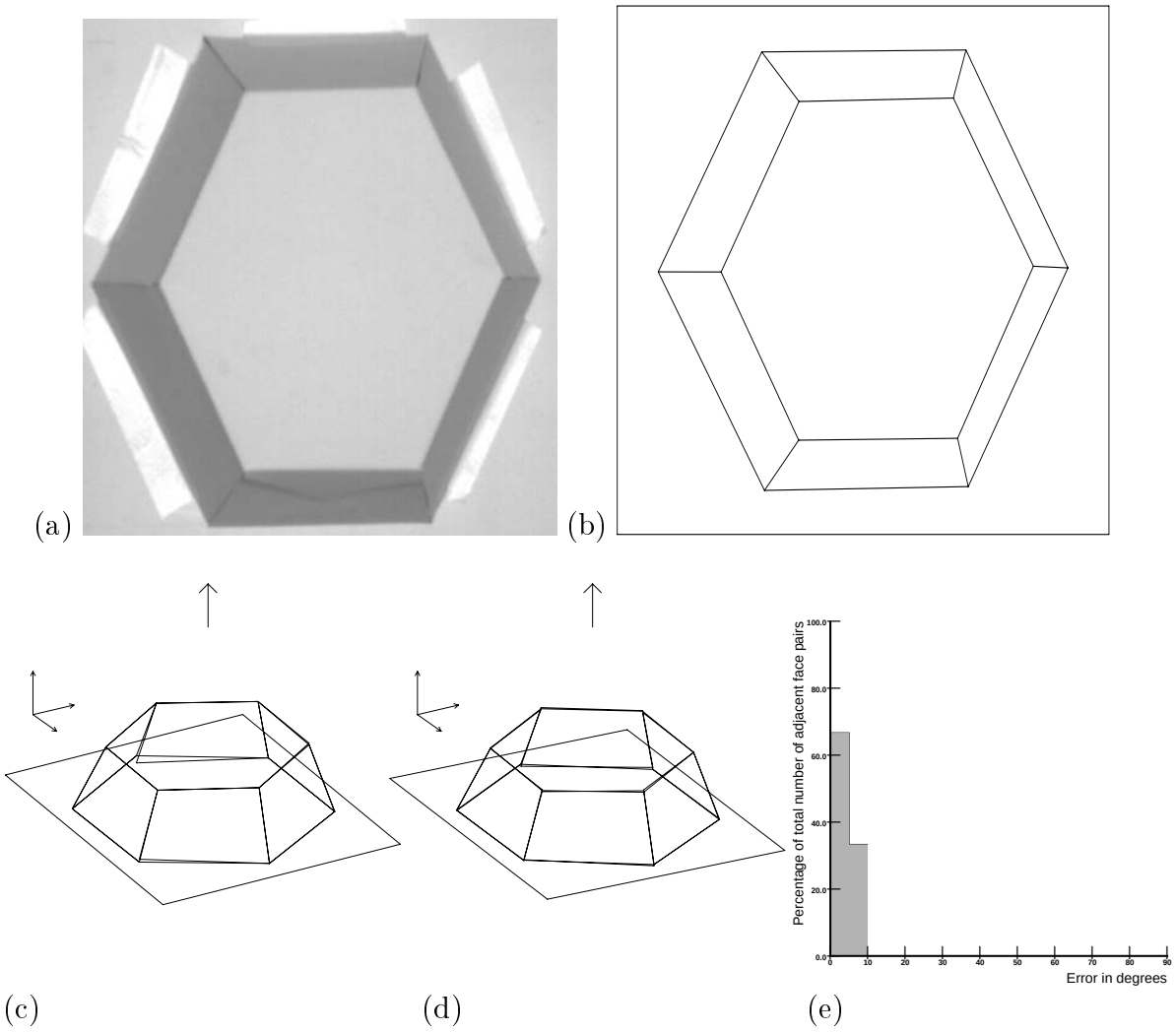


Figure 2.16 Results of the Lambertian model algorithm: (a) the input image; (b) the line-drawing; (c) first stage; (d) final stage of shape recovery; (e) the errors in the angles between adjacent faces presented in a histogram.

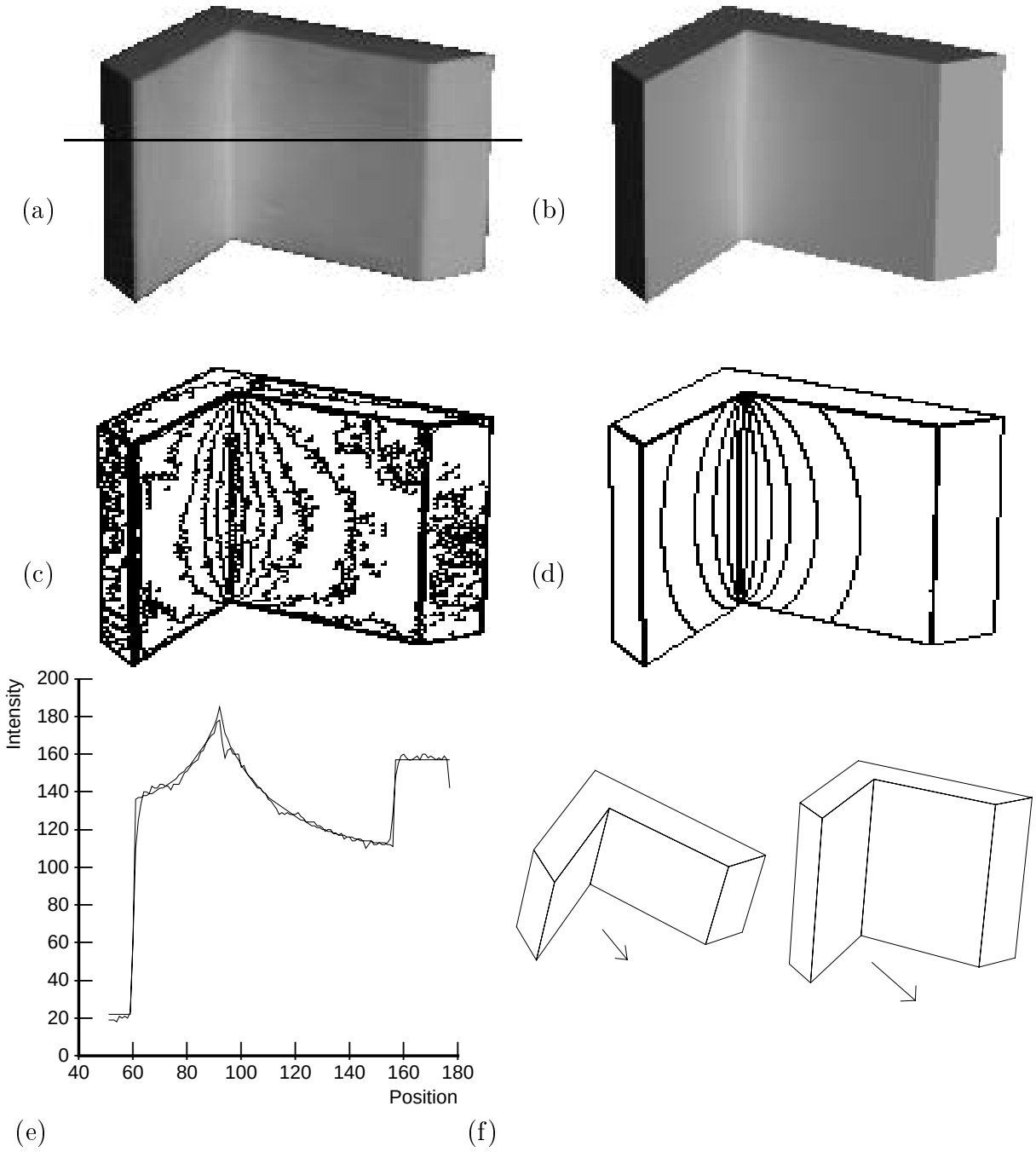


Figure 2.17 An L-shaped object: (a) the input image; (b) rendering of the recovered object; (c) iso-intensity lines in the image; (d) iso-intensity lines in the rendered image; (e) intensity values of the cross-section depicted by the line in (a) of the image and the rendered image; (f) views of the recovered object.

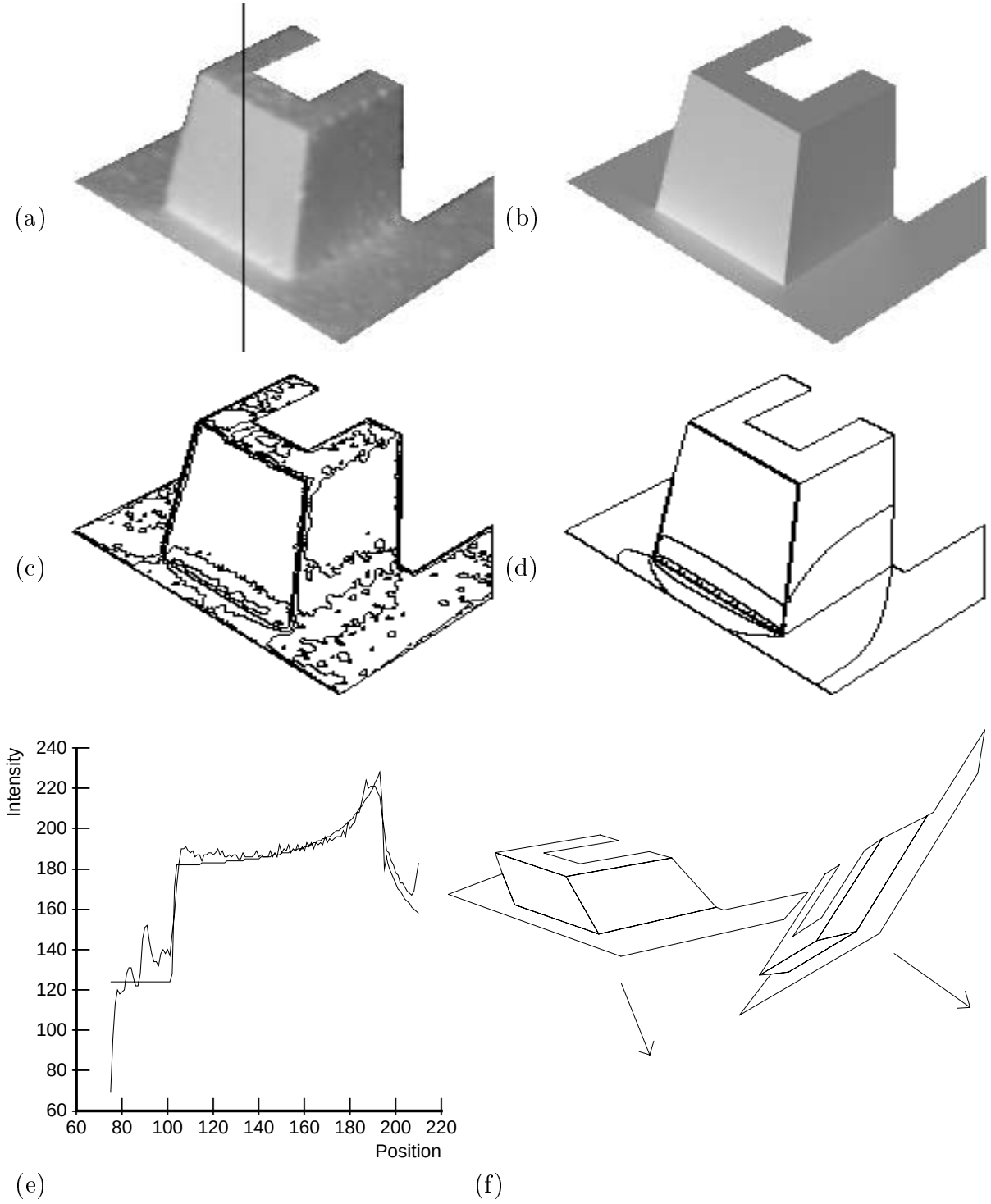


Figure 2.18 An image of a styrofoam object: (a) the image of the object; (b) rendering of the recovered object; (c) iso-intensity lines in the image; (d) iso-intensity lines in the rendered image; (e) intensity values of the cross-section depicted by the line in (a) of the image and the rendered image; (f) views of the recovered object.

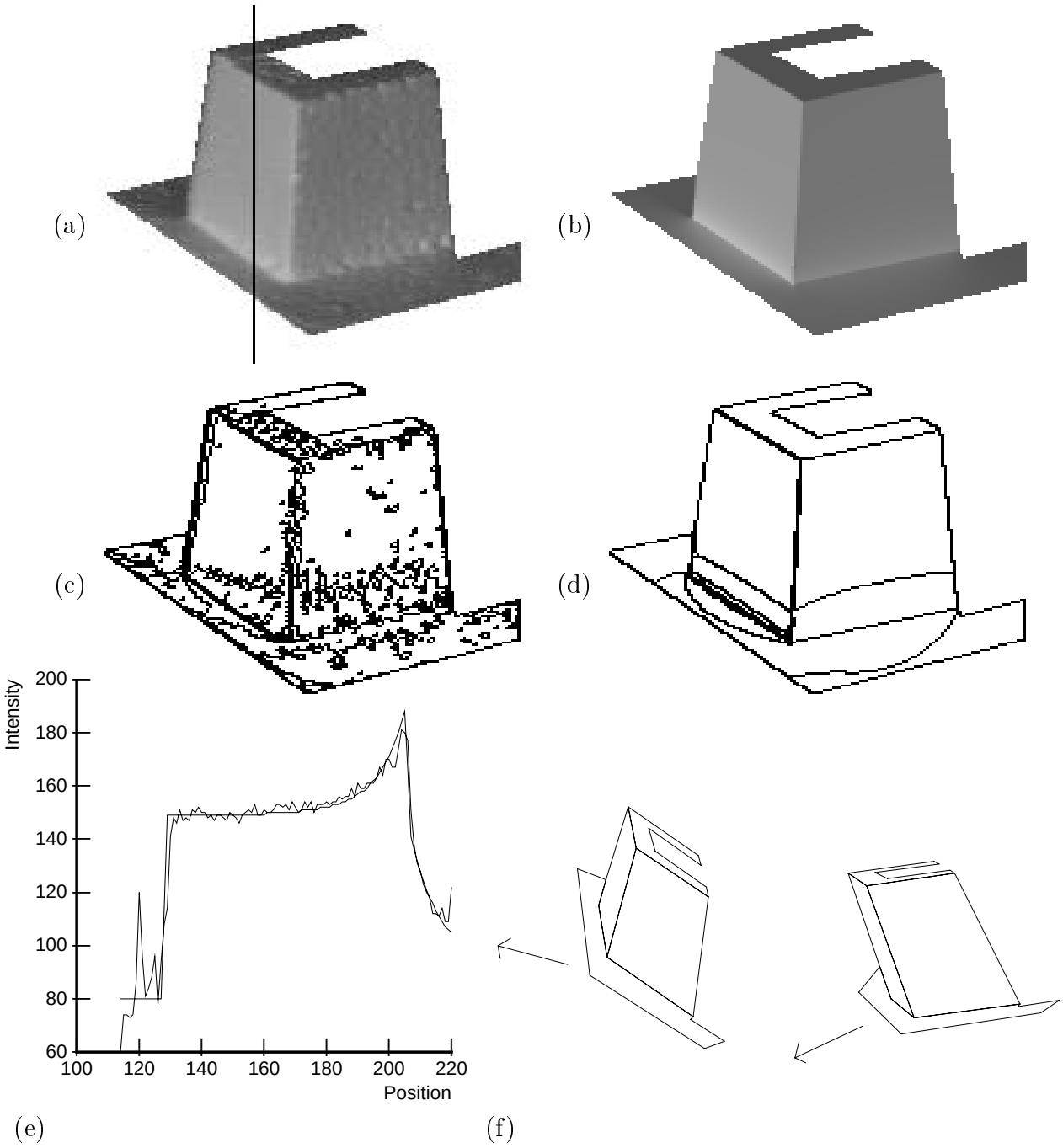
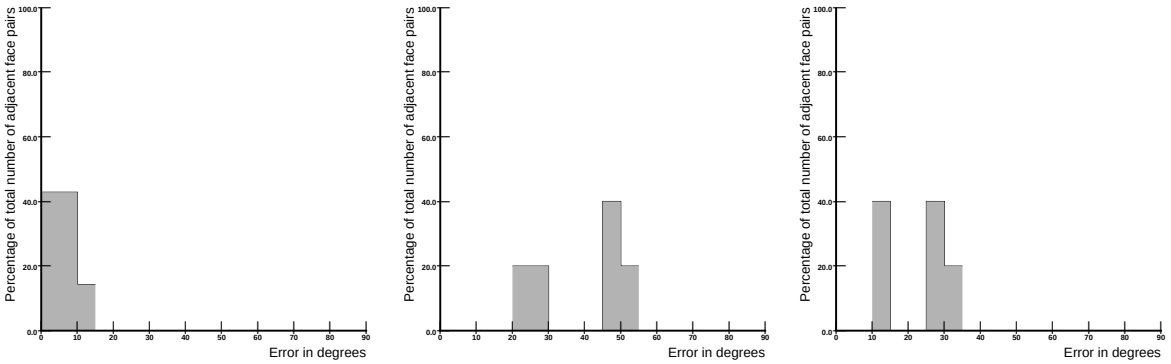


Figure 2.19 Second image of the styrofoam object: (a) the image of the object; (b) rendering of the recovered object; (c) iso-intensity lines in the image; (d) iso-intensity lines in the rendered image; (e) intensity values of the cross-section depicted by the line in (a) of the image and the rendered image; (f) views of the recovered object.



(a) (b) (c)

Figure 2.20 Error histograms comparing angles between adjacent faces measured in the object and measured in the recovered shape for the following objects: (a) the object in Figure 2.17; (b) the object in Figure 2.18; (c) the object in Figure 2.19.

2.6.4 Specular Objects

We have also tested our shape recovery algorithm on specular objects. Figure 2.21(a) shows the image of an object painted with glossy white paint. Notice that one of the faces displays a specularity. The algorithm was able to take advantage of the fact that one of the faces was specular by calculating directly an estimate for the direction of the light source when given an estimate for the parameters of the specular face, thus reducing the number of unknowns and speeding up the shape recovery process. Figure 2.16(b) shows the line-drawing extracted by hand from the edge detection results. Figure 2.16(c) shows two views of the reconstructed object. The arrows in Figure 2.16(c) show the direction from which the image was taken. In order to obtain a quantitative evaluation of our results we compared the angles between adjacent faces in the original objects and in the recovered object. The error measured was between 0.1° and 15° and the average was 6°. The distribution of the errors is shown in the histogram in Figure 2.16(d). We consider these results to be quite good taking into account that the reflectance model does not model exactly the reflectance of the object.

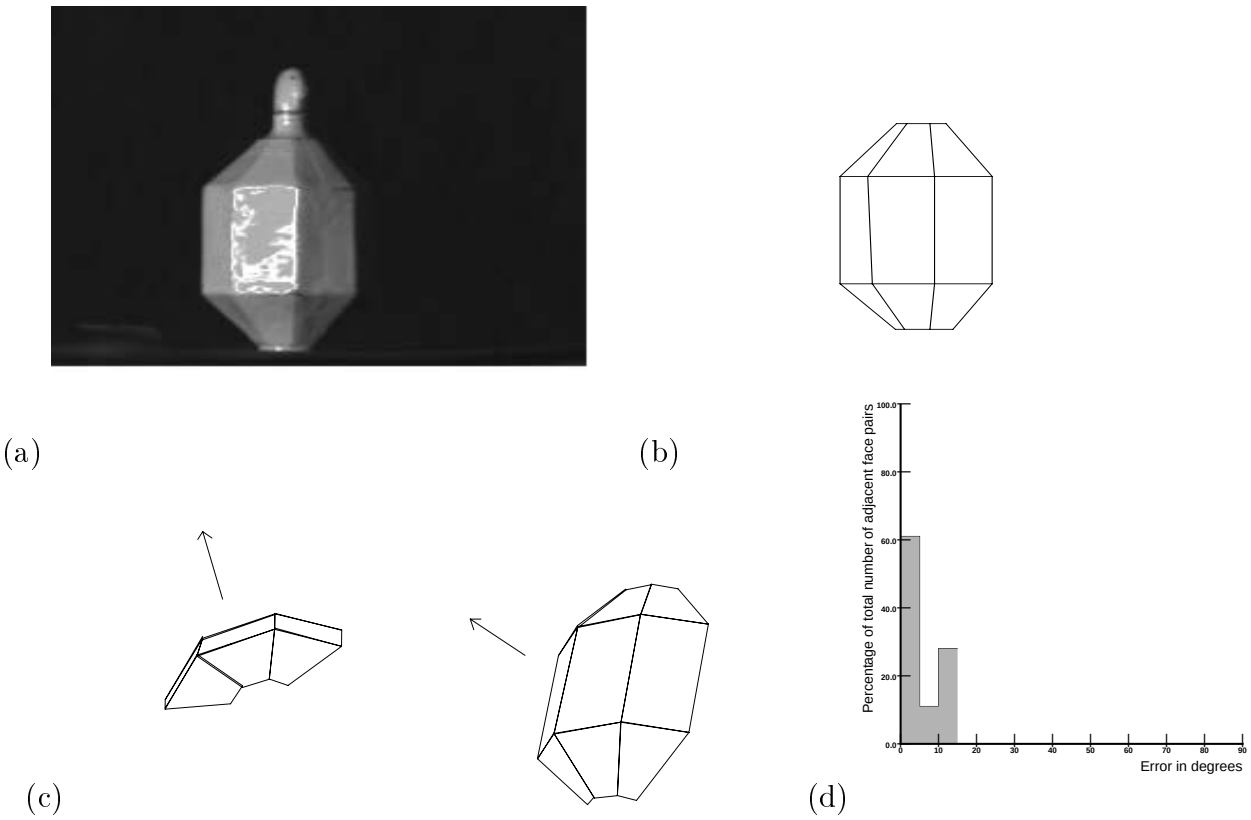


Figure 2.21 Results for the model for specular objects: (a) the input image; (b) the line-drawing; (c) two views of the recovered shape; (d) the errors in the angles between adjacent faces presented in a histogram.

2.7 Discussion and Future Work

We have presented a new approach to 3D shape recovery from a 2D image using the geometric information contained in the line-drawing and the shading information contained in the image. The geometric constraints imposed by the line-drawing of a polyhedron are represented by a set of linear equalities and inequalities, and the uncertainty in vertex position is explicitly taken into account.

We have shown that this scheme can be used to recover the 3D structure of objects which have complex reflectance models. It can actually be used for any polyhedral object whose reflectance can be reliably modelled. This includes images with shadows, metallic

or dielectric objects, rough or smooth objects etc... Reflectance parameters (such as albedo or roughness) and light source parameters can be recovered as part of the process.

CHAPTER 3

Finite-Resolution Aspect Graphs of Polyhedral Objects

3.1 Introduction

We address the problem of taking image resolution into account during the construction of the aspect graph [54] of a polyhedral object. Work in this area was pioneered by Kender and Freudenstein [51] and by Eggert et al. [25]. We focus on the case of an orthographic camera which cannot resolve image points closer than some preset distance and present a full implementation of an algorithm for computing the finite-resolution aspect graph of a (not necessarily convex) three-dimensional polyhedron.

We believe that this is an important problem for two reasons. First, the size of aspect graphs increases very fast as a function of object complexity. For a polyhedron with n faces, the optimal data structure presented in [34] requires $O(n^6)$ space for storing the aspect graph. For an algebraic surface of degree d, the best bound on the size of the aspect graph established so far is $O(d^{12})$ [68, 69, 75]. The time complexity of the algorithms for constructing the aspect graph is at least as high [34, 35, 69, 70]. In practice, this means that one can only use exact aspect graphs of relatively simple objects in tasks such as object recognition. However by taking resolution into account we can

hopefully simplify the aspect graph by disregarding features which cannot be seen due to the finite-resolution of the camera.

Second, we believe that an even more severe problem is that classical aspect graphs fail to correctly predict the appearance of objects in actual images. In a sense, they are not complex enough: "accidental" views that are in theory unobservable are in fact routinely observed because the finite resolution of the camera blurs several image features into a single one [51].

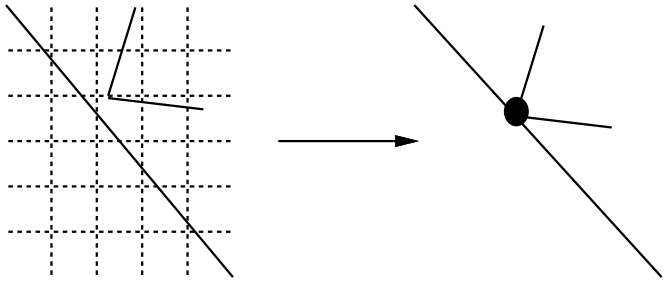


Figure 3.1 The effect of finite-resolution on an image. An edge and a vertex which do not touch each other in the infinite-resolution image seem to touch each other when finite-resolution is assumed.

We assume orthographic projection and suppose that two image points separated by a distance smaller than a preset threshold cannot be resolved. This is an appropriate model for a finite-resolution camera observing objects from a roughly constant distance. It is much more limited than the full-perspective, multi-scale projection model used in [25], but it affords a practical algorithm for non-trivial three-dimensional polyhedral objects. The effect of resolution is illustrated in Figure 3.1 where the distance between an edge and a vertex is below the resolution of the camera, and so these features seem to touch in the finite-resolution image.

The rest of the chapter is organized as follows. Section 3.2 discusses previous methods for aspect graph computation and introduces our approach. In Section 3.3, we present a catalogue of visual events adapted to our imaging model. It is an extension of the familiar EV-EEE events used by others. We then present in Section 3.4 an algorithm

for computing the finite-resolution aspect graph of a polyhedron. This algorithm uses homotopy continuation [61] which is described in Appendix B. An algorithm for simplifying the aspects by merging regions with identical finite-resolution aspect graphs is presented in Section 3.5. We discuss the size of the finite-resolution aspect graph and the complexity of the algorithm in Section 3.6. The implementation of the algorithm and results are presented in Section 3.7. Finally, a number of issues raised by our algorithm and its implementation are discussed in Section 3.8.

3.2 Literature Review

Informally, the aspect graph [54] is a qualitative, viewer-centered representation which enumerates all possible appearances of an object. More formally, choosing a camera model and a viewpoint determines the aspect of an object, i.e., the structure of the observed line-drawing. The range of possible viewpoints can be partitioned into maximal connected sets (regions) that yield identical aspects. The change in aspect at the boundary between regions is called a visual event. The maximal regions and the associated aspects form the nodes of an aspect graph, whose arcs correspond to the visual event boundaries between adjacent regions.

Since their introduction by Koenderink and Van Doorn [54] more than fifteen years ago, aspect graphs have been the object of very active research. Most of it has focused on polyhedra: indeed, several algorithms have been proposed for computing the exact aspect graph of these objects under orthographic [17, 35, 70, 76] and perspective [83, 84, 90, 91] projection. Some of these algorithms have actually been implemented (e.g., [70, 76, 83, 84, 90, 91]). Recently, algorithms for constructing the exact aspect graph of curved objects such as solids bounded by quadric surfaces [18], solids of revolution [24],

and algebraic surfaces [52, 69, 74, 82] have also been proposed and several of them have also been implemented.

Previous approaches assume that the object of interest is observed with a camera having infinite resolution. Here, we address the case of a camera having finite resolution [25, 51]. We say that an edge is observable if its visible portion projects onto a segment of length greater than some fixed ϵ. Similarly two vertices are distinct when the distance between their projections in the image is greater than ϵ.

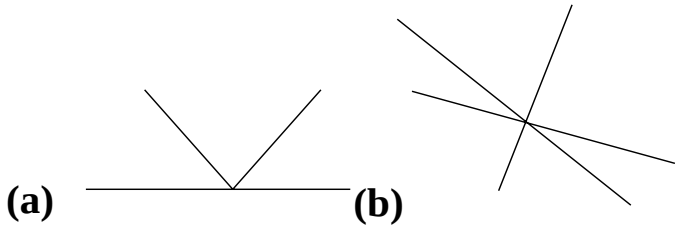


Figure 3.2 Infinite-resolution visual events: (a) an EV event; (b) a EEE event.

For infinite-resolution cameras, the appearance of a polyhedral object changes at a set of visual events where two types of coincidental alignments occur [14, 35, 70, 84]: EV events, where a vertex projects onto the image of an edge, and EEE events, where the projections of three edges intersect at a single point (Figure 3.2). In the perspective projection model, every point in an infinite 3D space represents a possible viewpoint. In this model, points from which the critical events are viewed form 2D surfaces. Under the orthographic projection model we assume that we view the object at infinite distance. Thus the viewing rays can be considered as points on a viewing sphere at infinity. The intersection of that sphere with the surfaces of the critical events yields a set of critical curves. Consider the example of Figure 3.3: for a certain viewing direction a vertex is projected on one side of an edge (Figure 3.3(a)). As the viewing direction continually changes the projection of the vertex will coincide with the projection of the edge (Figure 3.3(b)) and later be projected on the other side of the edge (Figure 3.3(c)). For each

point on the edge there is a viewing direction for which its projection and the projection of vertex coincide. Combining these viewing directions for all the points on the edge yields a critical curve.

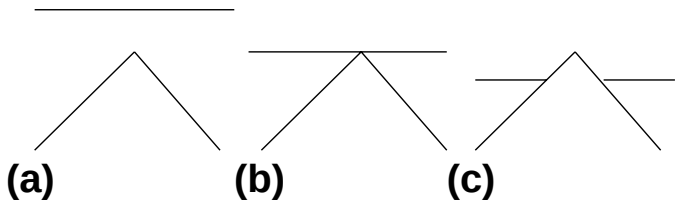


Figure 3.3 An EV event: (a) before the event; (b) at the event (c) after the event.

By taking resolution into account, we will introduce new types of visual events related to EV and EEE events. They will correspond to two vertices being so close they seem to coincide, a vertex seeming to touch an edge (similar to an EV event), and to foreshortening of an edge due to occlusion by two edges (similar to a EEE event). At the same time, "accidental" views such as triple junctions of occluding edges will survive over finite areas of the view space. We will partition the viewing sphere into non-critical regions bounded by critical curves using a plane-sweep algorithm [72]. For each region a representative view of the object is handed to a simplification algorithm, which merges close features generating the finite-resolution view of the object. Neighboring regions with identical finite-resolution views re merged producing the finite-resolution aspect graph.

3.3 Finite-Resolution Visual Events

We investigate what happens to the visual events of an infinite-resolution aspect graph under the assumption of a finite-resolution camera. Figure 3.4 shows the four new visual events. We derive implicit equations and a parametric representation for these curves. The corresponding viewing directions are parameterized such that each viewing direction has the same parameter value as a corresponding point on an edge. For example for the

EV curve each point on the edge has the same parameter value as the viewing direction on the curve where the projection of the vertex is closest to the projection of that point. The implicit equations are used to find intersection points between curves and the parametric representations are used to trace curves, find end-points on these curves corresponding to end-points of the edge, and order the points on each curve by their parameter value.

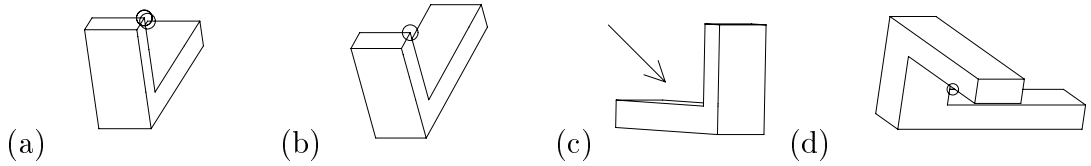


Figure 3.4 Four visual events: (a) a VV event; (b) an EV event; (c) a face begins to appear; (d) a EEE event.

3.3.1 Vertex-Vertex (VV) Event

When two vertices are viewed from a direction parallel to the line joining them, their projections coincide in the image. Changing the viewing direction increases the distance between the projections of the vertices, until it exceeds the camera resolution ϵ and the vertices are seen as distinct. The visual event occurs when the image distance is exactly ϵ. The same phenomenon happens for a viewing direction parallel to the direction of an edge. At that viewing direction, both ends of the edge coincide and the edge is foreshortened to zero. When the viewing direction changes and the length of the projected edge is more than ϵ, the edge becomes visible. In both cases, when the distance between the projections of the vertices is less than ϵ, they appear to coincide in the image and the adjacent edges seem to emanate from the same point.

Let $\mathbf{v}$ denote the viewing direction. The vertices p and q are represented by the pair $(p, \mathbf{u})$, where $q = p + \mathbf{u}$. The vertices are observable as distinct vertices if and only if $|\mathbf{u} \times \mathbf{v}| \geq \epsilon |\mathbf{v}|$ (Figure 3.5).

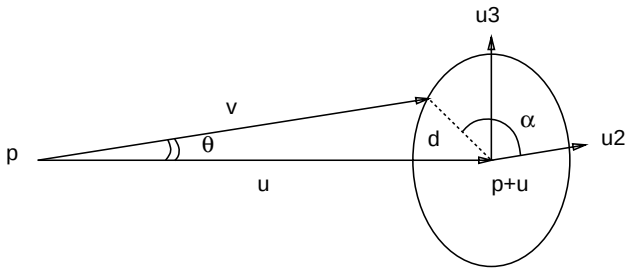


Figure 3.5 Illustration of the VV event.

The corresponding visual events correspond to views such that the angle between $\mathbf{v}$ and $\mathbf{u}$ is $\theta = \arcsin(\epsilon/|\mathbf{u}|)$. They form opposite small circles on the viewing sphere, which can be parameterized as follows: for each angle $0 \leq \alpha \leq 2\pi$ we compute a point that when subtracted from p produces the desired viewing direction. As shown in Fig 3.5, a parameterization of one of these circles is :

$$\mathbf{v}(\alpha) = \mathbf{u} + d(\cos\alpha\mathbf{u}_2 + \sin\alpha\mathbf{u}_3) \quad \alpha \in [0, 2\pi],$$

where $\mathbf{u}, \mathbf{u}_2, \mathbf{u}_3$ is an orthonormal basis of $\mathbb{R}^3$ and $d = \mathbf{u}\tan\theta$. The opposite circle is $-\mathbf{v}(\alpha)$.

3.3.2 Edge-Vertex (EV) Event

With infinite-resolution cameras, an EV event occurs when the projection of a vertex lies on the projection of an edge. With a finite-resolution camera, the vertex appears to be touching the edge when the distance from the projection of the vertex to the projection of the edge is less than ϵ (Figure 3.6). There are two cases: if the vertex is in front of the face containing the edge, the infinite-resolution event is replaced by two new events (Figure 3.6(a-d)). If, on the other hand, the vertex is behind this face, we can not see the vertex when it is below the edge because it is occluded, therefore we will have one infinite-resolution visual event plus one curve corresponding to the vertex being at a distance ϵ above the edge (Figure 3.6(e-g)).

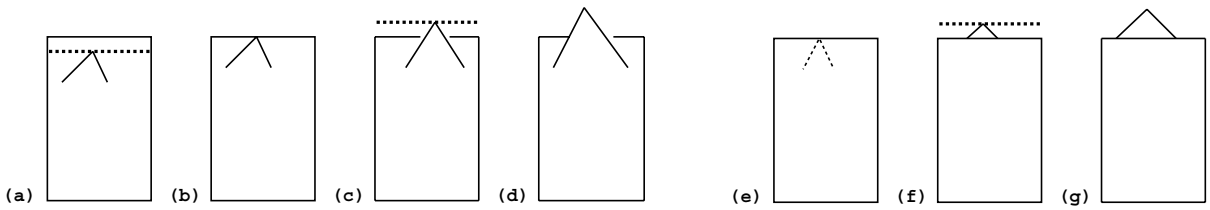


Figure 3.6 finite-resolution EV event. When the vertex is in front of the edge: (a) first finite-resolution EV event; (b) infinite-resolution EV event; (c) second finite-resolution EV event; (d) vertex completely above the edge. When the edge is in front of the vertex: (e) infinite-resolution EV event; (f) finite-resolution EV event; (g) vertex completely above the edge.

Consider an edge e with endpoints e_1 and $e_2 = e_1 + \mathbf{u}$ and a vertex p. A critical viewing direction is a vector $\mathbf{v}$ such that p is at distance ϵ from the plane that contains e_1 with normal $\mathbf{u} \times \mathbf{v}$. This condition can be written as:

$$(p - e_1) \cdot (\mathbf{u} \times \mathbf{v}) = \epsilon \mid \mathbf{u} \times \mathbf{v} \mid,$$

which is a quadratic equation in $\mathbf{v}$.

We take a different approach for constructing a parametric representation. For every viewing direction where the distance from the projection of the vertex p to the projection of the edge e is ϵ, there must be a point on e whose projection is closest to the projection of p. That distance must be ϵ. We parameterize the edge e by $e(t) = (1-t)e_1 + te_2$, with $0 \leq t \leq 1$. For each $e(t)$, we compute a viewing direction $\mathbf{v}(t)$ by finding a point $q(t)$ such that $\mathbf{v}(t) = p - q(t)$. $q(t)$ satisfies the following conditions which are illustrated in Figure 3.7:

$$\begin{cases} |q(t) - e(t)| = \epsilon, & \\ (q(t) - e(t)) \perp \mathbf{u}, & \text{since } e(t) \text{ is the closest point on } e \text{ to } q(t), \\ (q(t) - e(t)) \perp (p - q(t)), & \text{so that the distance in the image will be } \epsilon. \end{cases}$$

The first and third conditions characterize a circle on the sphere whose center is $e(t)$ and whose radius is ϵ. Intersecting that circle with the plane perpendicular to $\mathbf{u}$ which goes through $e(t)$ produces the two solutions of our problem. The above equation is adequate

for finding viewing directions where $e(t)$ is internal to the edge. However, for the endpoint of the edge the second condition is relaxed. The result is a semi-circle (Figure 3.8). This semi-circle is part of the curve of the VV event we discussed in the previous section.

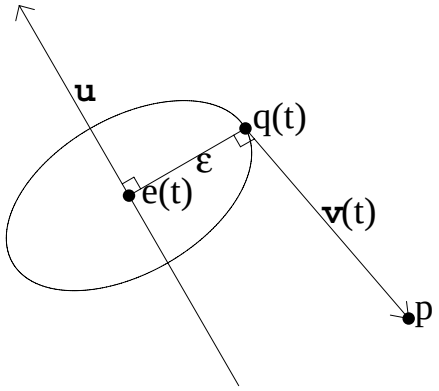


Figure 3.7 Parametric construction of the EV curve. Illustration of the geometric construction of the viewing direction $\mathbf{v}(t)$ corresponding to the point $e(t)$ on the edge.

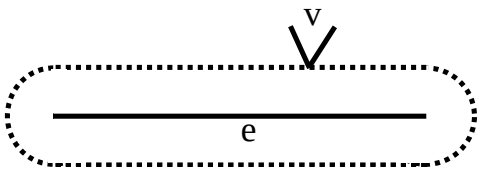


Figure 3.8 The positions where the projection of the vertex can be with respect to the edge.

It should also be noted that, for infinite-resolution aspect graphs, EV events for which the vertex and the edge belong to the same face are normally not considered. Instead, special events corresponding to the appearance of a face when the viewing direction crosses the face's plane are considered separately. Here, this case is included in the EV events, and a face appears when the distance between the projection of at least one of its edges to one of its vertices becomes longer than ϵ (Figure 3.9).

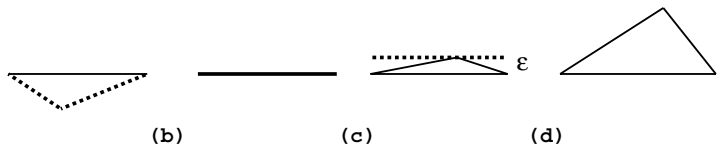


Figure 3.9 (a) The face is not visible. (b) Infinite-resolution face-appearing event. (c) The finite-resolution face-appearing event. (d) Face completely visible.

3.3.3 Edge-Edge-Edge (EEE) Event

Beside foreshortening, an edge may also seem to disappear because occlusion reduces its visible projected length to ϵ. This happens when two faces occlude the edge e and the distance between the intersection points of the corresponding edges e_1 and e_2 with e is less than ϵ. In the image produced by a finite-resolution camera all three edges appear to be intersecting at a point. This is the triple edge intersection event known as EEE (Figure 3.10). We now characterize the views $\mathbf{v}$ such that the projection of the edges e_1 and e_2 intersect the projection of the edge e at a distance ϵ from each other.

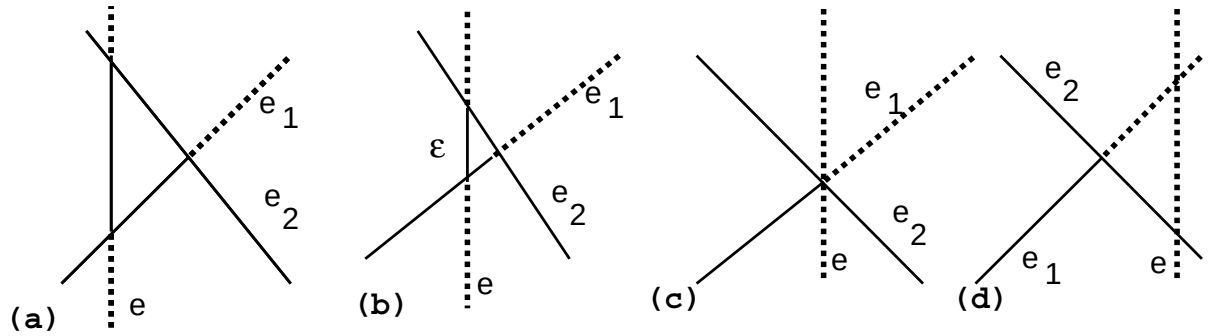


Figure 3.10 Finite-Resolution EEE event: (a) the three edges are visible; (b) the finite-resolution EEE event; (c) the infinite-resolution EEE event; (d) edge completely occluded.

To characterize intersections between straight lines in a simple way, we use Plücker coordinates, which describe a line by two orthogonal vectors (a, b) where a is some direction along the line, and if p is a point on the line, $b = p \times a$. We use the following property: two straight lines with Plücker coordinates (a_1, b_1) and (a_2, b_2) intersect if and only if $a_1 \cdot b_2 + a_2 \cdot b_1 = 0$. We write that a line D_1' passing through $e(t) = p + t\mathbf{u}$ with direction $\mathbf{v}$ intersects the supporting line $D_1 = (a_1, b_1)$ of e_1. We choose the origin in p, and the line D_1' has Plücker coordinates $(\mathbf{v}, t\mathbf{u} \times \mathbf{v})$. Similarly, we write that a line D_2' passing

through $p + (t + \epsilon'/|\mathbf{u}|)\mathbf{u}$ with direction $\mathbf{v}$ intersects the supporting line $D_2 = (a_2, b_2)$ of e_2 (see Figure 3.11). The line D_2' has Plücker coordinates $(\mathbf{v}, (t + \epsilon'/|\mathbf{u}|)\mathbf{u} \times \mathbf{v})$. Writing that D_1' (resp. D_2') intersects D_1 (resp. D_2), we obtain two equations:

$$\begin{cases} ta_1 \cdot (\mathbf{u} \times \mathbf{v}) + b_1 \cdot \mathbf{v} = 0, \\ (t + \epsilon'/|\mathbf{u}|)a_2 \cdot (\mathbf{u} \times \mathbf{v}) + b_2 \cdot \mathbf{v} = 0, \end{cases} \tag{3.1}$$

and eliminating t among these yields:

$$\epsilon' = \frac{b_1 \cdot \mathbf{v}|\mathbf{u}|}{a_1 \cdot (\mathbf{u} \times \mathbf{v})} - \frac{b_2 \cdot \mathbf{v}|\mathbf{u}|}{a_2 \cdot (\mathbf{u} \times \mathbf{v})} = \epsilon \frac{|\mathbf{v}||\mathbf{u}|}{|\mathbf{u} \times \mathbf{v}|}, \tag{3.2}$$

where ϵ' is the actual distance between the two intersection points on e. Squaring this equation yields a homogeneous equation of degree 6 in $\mathbf{v}$.

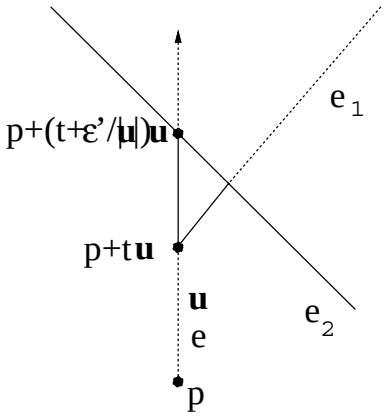


Figure 3.11 Illustration of the construction for the finite-resolution EEE event equation: The projection of points $e(t) = p + t\mathbf{u}$ and $p + (t + \epsilon'/|\mathbf{u}|)\mathbf{u}$ intersect the projections of e_1 and e_2 respectively.

We parameterize the curve using the parameter t used in (3.1). At viewing direction $\mathbf{v}(t)$ the point $e(t)$ on e intersects the projection of e_1, and the point $e(t) + (\epsilon'/|\mathbf{u}|)\mathbf{u}$ intersects the projection of e_2. Using

$$\epsilon' = \epsilon \frac{|\mathbf{v}(t)||\mathbf{u}|}{|\mathbf{u} \times \mathbf{v}(t)|}, \tag{3.3}$$

and (3.1) we recover $\mathbf{v}(t)$ using homotopy continuation [61] (see Appendix B for a description of this method). An alternative method for finding $\mathbf{v}(t)$ is to find ϵ' which

satisfies (3.3) using standard numerical one-dimensional optimization techniques such as a combination of the secant method and the Newton-Raphson method [48]. The method requires that for a given value of ϵ' we evaluate the function being minimized (3.3). We substitute the value of ϵ' into (3.1), which yield two linear equations in $\mathbf{v}(t)$, which are readily solved. We then evaluate (3.3) using the result. This method is more efficient then the previous one because instead of solving for ϵ' and $\mathbf{v}(t)$ at the same time, we use efficient one-dimensional optimization techniques to solve only for ϵ' and use its value to compute $\mathbf{v}(t)$. We have not yet implemented this technique and use in our implementation a similar but less efficient variant of it.

3.4 Constructing the Aspect Graph

We have discussed the different critical events and their equations, we will now show how to generate the aspect graph. The algorithm is divided into the following steps:

1. Generating the visual event curves from their equations.
2. Finding intersection points between curves.
3. Eliminating events which cannot be seen due to occlusion.
4. Constructing the stable regions and the corresponding aspects.

3.4.1 Generating the Visual Event Curves

As suggested in [34], we use a cube with edges of length 2 to represent the view space. Given the parametric representation of the visual event curve derived earlier, we construct a discrete representation of the visual event curve by sampling the parameters. We compute a viewing direction for discrete values of the parameter, then normalize the viewing directions such that $|\mathbf{v}|_\infty = 1$.

3.4.2 Finding Intersections

We find intersection points between two curves using homotopy continuation. In this case, the implicit equations of both curves are given to the algorithm. As we use the viewing cube we have to find the intersection points on the six faces of the cube, therefore we run the algorithm six times, each time setting one of the coordinates of $\mathbf{v}$ to ± 1, reducing the number of unknowns by one. The algorithm returns the intersection points. Points such that $|\mathbf{v}|_\infty > 1$ are discarded. We compute the parameter value for the point and discard points whose value out is of range. We insert the remaining points into the correct place on the curve, using their parameter value.

3.4.3 Occluded Events

Until now, a viewing direction has been considered to be part of a critical curve when the features (edges, vertices) participating in the event satisfy the corresponding critical curve equation. However, in order for this viewing direction to be part of the actual critical curve, the participating features must all be visible. This requirement has a straightforward meaning for infinite-resolution events but for finite-resolution events a more delicate definition is needed. Consider the example of an EV event in Figure 3.12(a). The position on the edge which is at distance ϵ away from the vertex is not visible but as we can see other parts of the edge, we consider this viewing direction as part of the critical curve. On the other hand in Figure 3.12(b), the vertex and its adjacent edges are occluded therefore this event is not part of the critical curve. Therefore we also consider a point on the critical curve visible, even when some of the features occlude each other but features adjacent to the occluded feature are visible.

A number of tests must be performed to check for occlusion. First, the features participating in the event can be occluded by a face lying in front of them. The point on the curve where the event becomes occluded is the point where an edge of another face

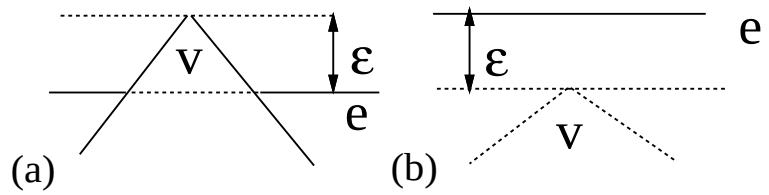


Figure 3.12 Occlusion of finite-resolution events: (a) although the point on the edge is occluded the event is considered visible because the other parts of the edge are visible; (c) the vertex and its adjacent edges are occluded therefore the event is not part of the critical curve.

starts to occlude the features. In an EV event this point also lies on the EV curve of the same vertex and the edge of the occluding face. Therefore the place on the curve where occlusion starts is the intersection point between the two EV curves. In the EEE event a similar phenomenon happens and the place where the occlusion starts is the intersection point between two EEE curves which share two of the three edges.

Second, the features participating in the event can also be occluded by faces adjacent to them. In this case the place on the curve where the event becomes occluded is the point where the curve intersects one of the EV curves where the edge and the vertex belong to the occluding face. So in all cases the occluded part of a curve starts and ends at intersection points with other curves, and we use the technique described above to recover these points.

3.4.4 Constructing the Regions and Aspects

We compute the regions from their bounding critical curves and curve intersections. For each face on the viewing cube we use a plane-sweep algorithm [72]. One of the important reasons for using the viewing cube instead of the viewing sphere was to enable us to use the standard plane-sweep algorithm on the faces of the cube. The algorithm was adapted for dealing with curves by adding critical events at extremal points, i.e., places where the curve is perpendicular to the sweeping direction [34]. Points where curves start, end or

intersect are processed in sweep order, generating the non-critical regions of the aspect graph (Figure 3.13). For each region the algorithm returns a point in it which is its representative viewing direction.

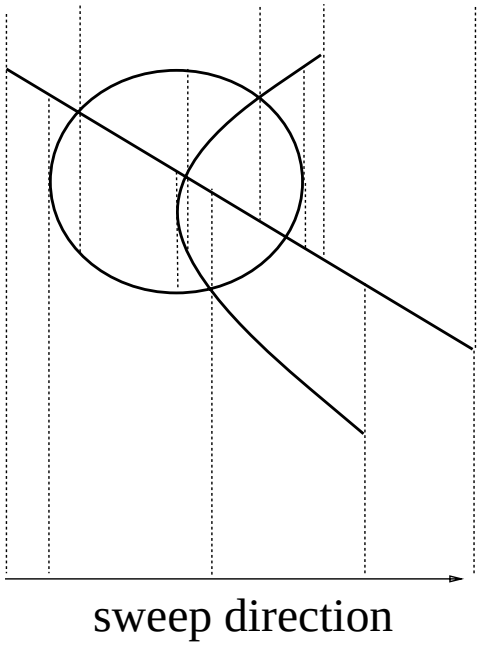


Figure 3.13 Illustration of the plane-sweep algorithm.

3.5 Simplifying The Aspect Graph

3.5.1 Finite-Resolution Aspects

The plane-sweep algorithm returns a representative viewing direction for each region in the finite-resolution aspect graph. For each such viewing direction we render the object as seen from that direction. We attempt to merge close features to generate a simpler view of the object, which we call the finite-resolution aspect.

When merging features we can not merge them in an arbitrary order because we have to take into account the fact that "nearness" is not transitive. To illustrate this let us present a few examples. In Figure 3.14(a), two edges which are close to each other

appear in the image as a thick edge. As the distance between the edges is less than ϵ the two edges are perceived as one line. In Figure 3.14(b) however, edge v is close to edge u and to edge w, but edges u and w are far from each other. The resulting image is a thick region which can not be simplified into a line as in the previous example, and we have to leave the black rectangle in the representation. If an edge is close to a vertex and transitivity of nearness is not violated the vertex is placed on the edge. however, in the example shown in Figure 3.14(c), vertex p is near edges e_1 and e_2 that are adjacent to vertex q. The result should be merging p and q, but p is not near q, violating the transitivity of nearness. Although the vertices are not close to each other we merge the edges and vertices. The resulting vertex is marked as a "thick vertex" because the image of that vertex will be thicker than a regular vertex.

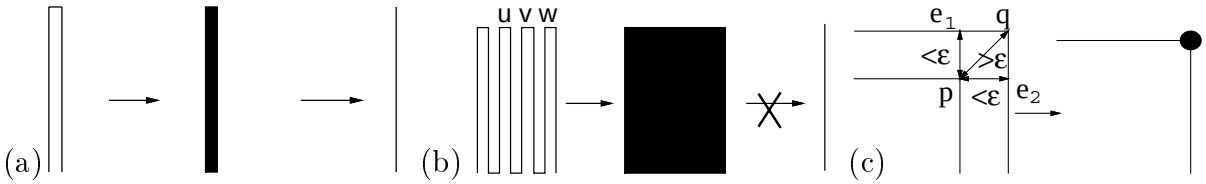


Figure 3.14 Examples of aspect simplification: (a) two close lines and their simplified image: (b) a comb-shaped object and its simplified image: (c) a vertex which is close to two edges but not to the vertex which is adjacent to them and the simplified image with a "thick vertex".

After simplifying the aspects of all the regions in the aspect graph, we compare aspects of neighboring regions, and regions with identical finite-resolution aspects are merged. The fact that adjacent regions would have identical aspects is a somewhat surprising phenomenon because we would expect different regions to have different aspects. Neighboring aspects may turn out identical because different active events can have the same simplifying effect. For example in Figure 3.15 an aspect in which a face has disappeared is produced from two neighboring aspects. These kinds of phenomena are very common and they reduce the size of the aspect graph considerably.

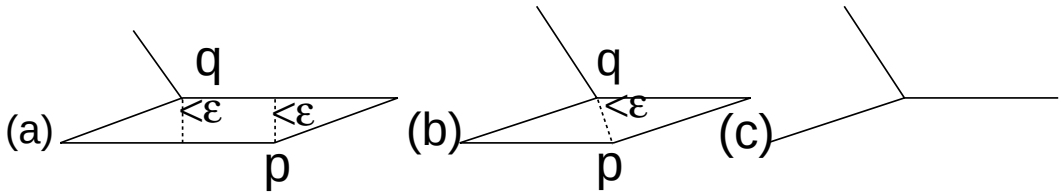


Figure 3.15 Two different aspects producing the same simplified aspect: (a) vertices p and q are near opposite edges; (b) vertices p and q are near each other and are near opposite edges; (c) The simplified aspect.

3.5.2 Aspect Simplification

In the initial stage of the algorithm, we build a graph representing incidence relationships between the edges and the vertices in the infinite-resolution aspect, and a list of pairs of features that are near each other. There are various operations which can be done to incorporate pairs of features into the graph. They can be as simple as merging two vertices or placing a vertex on an edge, or more complicated as creating a "thick vertex". For each such operation there are preconditions which must be fulfilled; changes are made to the graph, and the corresponding pairs of features are removed from the list. Complex operations were defined only when their effect could not be achieved by applying a series of simpler operations. Operations are sorted by complexity, and the most complex operation whose preconditions have been met is performed. The process is repeated until there are no more operations that can be performed. For a list of all simplification operations refer to Appendix A.

In Figure 3.16 a number of examples of aspect simplification are shown. In Figure 3.16(a) two close vertices are merged and the corresponding edge is removed. In Figure 3.16(b) a very narrow face is removed. Although a number of vertices and edges are close to each other the more complex operation of face removal is performed. Figure 3.16(d) is a more complicated example of the same operation. In Figure 3.16(c), two sets of close edges are merged into only two edges, and in Figure 3.16(e) an example of the "thick vertex" and two face removal operations is shown.

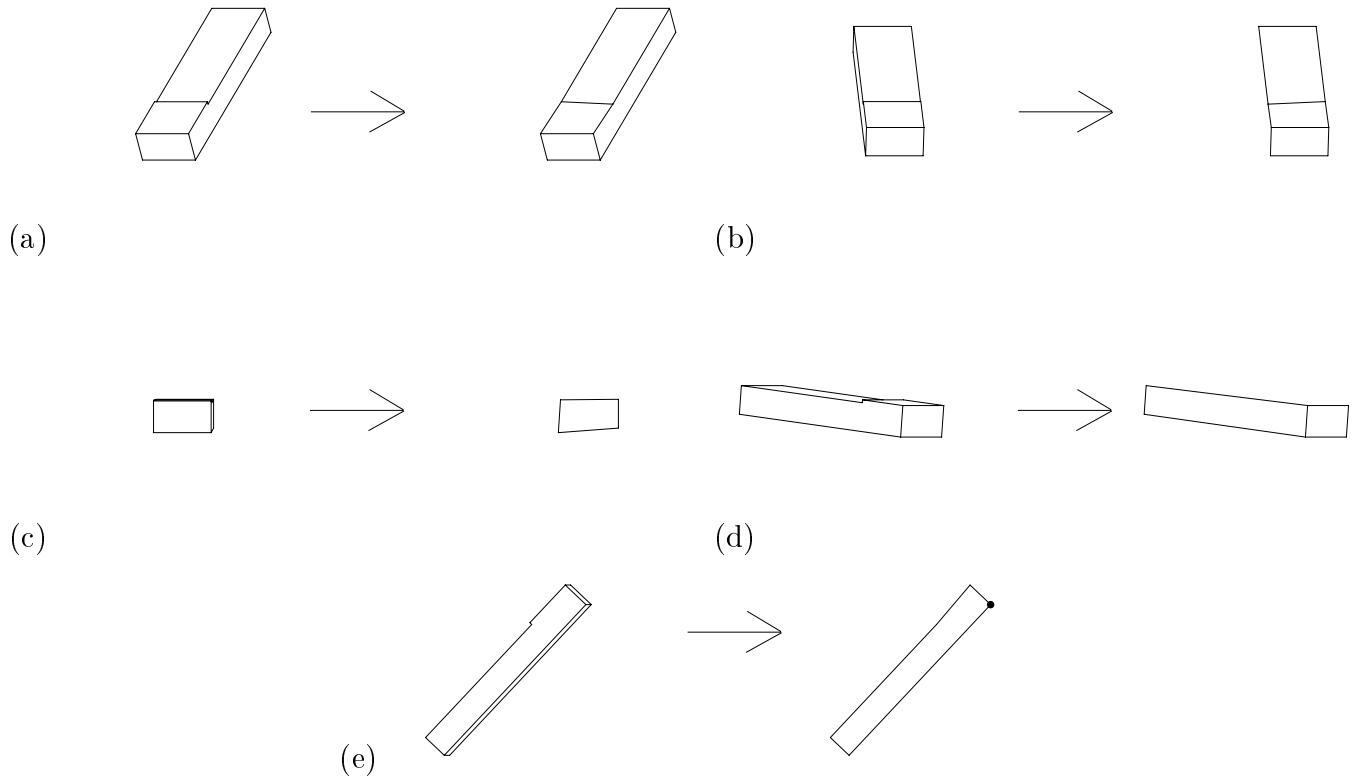


Figure 3.16 Examples of infinite-resolution aspects and their simplified aspects: (a) a short edge removed; (b) a narrow face is removed; (c) many close edges merged; (d) narrow faces removed; (e) a "thick" vertex created.

In the last stage of the algorithm, finite-resolution aspects of neighboring regions are compared. Identical aspects are merged. In order to merge two regions the two graphs have to be isomorphic and the vertices from the infinite-resolution aspect which appear in the finite-resolution aspect have to map to the same vertices in the finite-resolution aspects.

3.6 Size of The Aspect Graph and Algorithm Time Complexity

It was shown in [70] that the complexity of the aspect graph for a polyhedral object of size n is $O(n^6)$. We use a similar argument to show that the size of the finite-resolution aspect graph is also $O(n^6)$. There are $O(n^2)$ vertex-vertex and edge-vertex curves and $O(n^3)$ edge-edge-edge curves. By Bezout's theorem the number of intersection points between several polynomials is the product of their degrees; since the degree of the implicit equations of the curves is bounded by a constant (i.e., 6), a pair of curves will intersect on the viewing sphere at most a constant (i.e., 72) number of times. Therefore each curve will be intersected by other curves in $O(n^3)$ places yielding $O(n^3)$ edges and vertices. For all curves there will be $O(n^6)$ edges and vertices. The number of regions which is also $O(n^6)$ is found using Euler's formula $|F| = 2 - |V| + |E|$, where $|E|, |V|$ and $|F|$ is the number of edges vertices and regions respectively. Because there are more finite-resolution curves than infinite-resolution curves and the degree of the implicit equations of the finite-resolution curves is higher, the actual size of the finite-resolution aspect graph will usually be larger but the asymptotic bound is still the same.

We use an argument similar to the one in [34] to analyze the complexity of the algorithm for computing the finite-resolution aspect graph. The time required to generate the $O(n^3)$ curves is $O(n^3)$. For each curve we determine which segments of the curve represent views in which the event is visible by finding the points on the curve where the features participating in this event become occluded by an edge. Therefore each of the $O(n^3)$ curves is intersected with $|E| = O(n)$ curves yielding $O(n^4)$ intersection points at a cost of $O(n^4)$. Sorting the intersection points and determining the actual segments of visible event curves has the complexity of $O(n^4 \log n)$. As described above the number of vertices, edges and regions in the aspect graph is $O(n^6)$, however most aspect graphs are

much smaller. For an aspect graph of actual size m, the time for running the plane-sweep algorithm for constructing it is $O(m \log m)$.

The overall time complexity for tracing the curves and finding the regions of the aspect graph is $O(n^4 \log n + m \log m)$. The complexity of generating the aspect graph is the same as for the algorithm to generate the infinite-resolution aspect graph described in [34]. However our algorithm has an additional simplification stage which has to be run on a representative of each of the m regions of the finite-resolution aspect graph. In that stage the object must be backprojected and "active" events must be found and processed. This stage whose time complexity is at least $O(m\ n)$ is larger than the complexity of the previous stages and dominates the complexity of algorithm.

3.7 Results

In this section we present results obtained by running the algorithm on several objects. the infinite-resolution visual event curves or the object shown in Figure 3.17(a) are shown in Figure 3.17(b); there are 12 different regions in this case (compare to [34]). The visual event curves of the finite-resolution aspect graph are shown in Figure 3.17(c). These curves bound 377 different regions. The aspect graph is shown in Figure 3.17(d): adjacent regions yielding the same aspect have been merged, and the resulting 117 regions are shown in different shades of grey.

Figure 3.18 shows the qualitatively different aspects obtained by making a continuous sweep across some of the regions of the aspect graph. This sweep is indicated by the black diagonal line in Figure 3.17(d). The viewing direction corresponding to each aspect is indicated by a small circle. The aspects in Figure 3.18 have been rendered by merging features (vertices and/or edges) which are closer than ϵ. "Accidental" aspects now exist over a finite area of the viewing cube (e.g., Figure 3.18.(i)). In other aspects (e.g., Figure

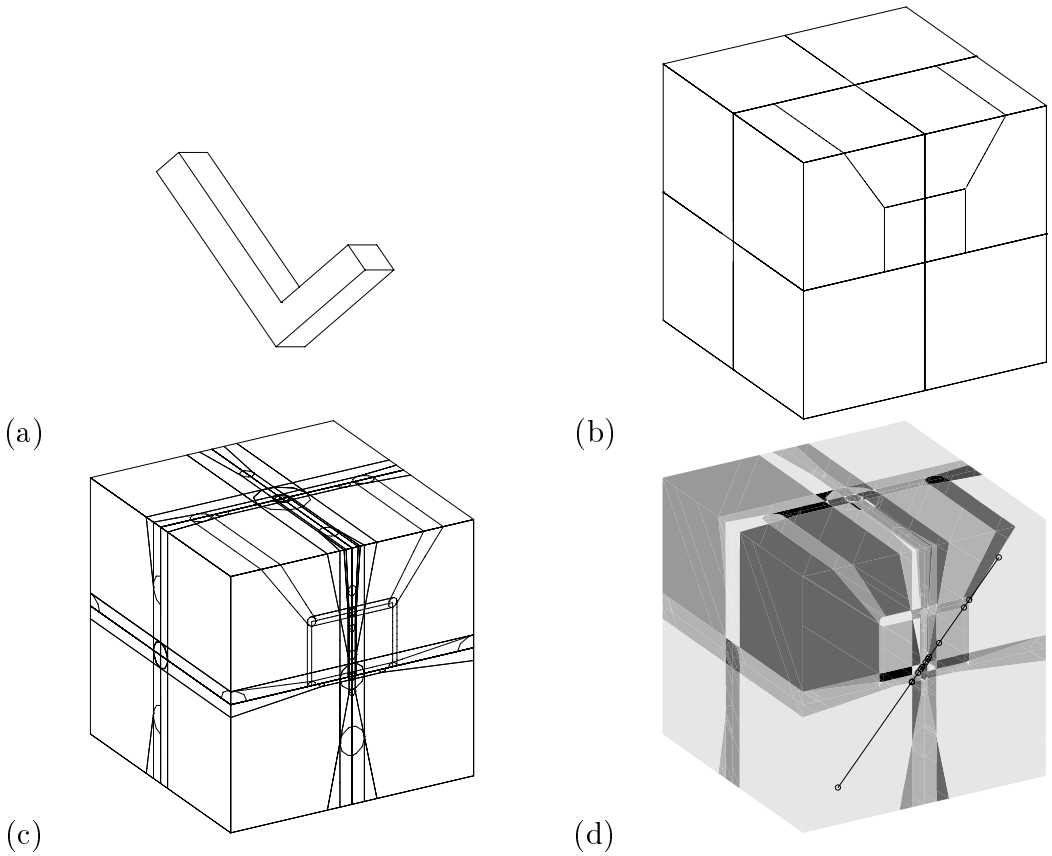


Figure 3.17 An L-shaped object and its aspect graph: (a) the object; (b) the infinite-resolution visual event curves; (c) the finite-resolution visual event curves; (d) the finite-resolution aspect graph.

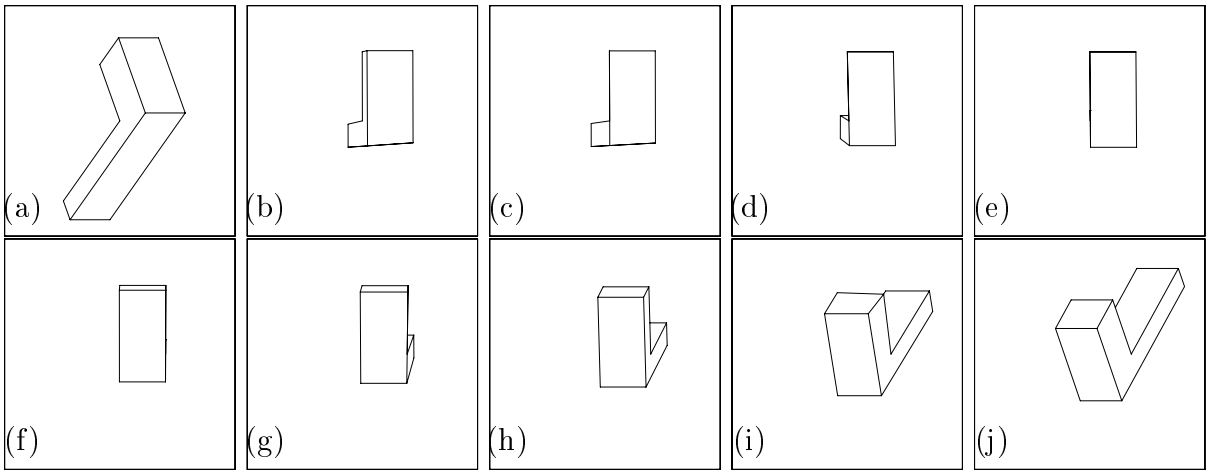


Figure 3.18 A series of aspects obtained by sweeping through adjacent views in the aspect graph.

3.18.(c,d)), the image does not look like a real aspect of the object because the finite-resolution aspect is intended to convey relationships between edges and vertices and not the image of the object itself.

Figure 3.19 shows the results obtained by running the algorithm on a more complex object (Figure 3.19(a)) which yields EEE curves. The visual event curves of the finite-resolution aspect graph are shown in Figure 3.19(b). The EEE curves are shown as heavy lines. The aspect graph shown Figure 3.19(c) contains 357 regions.

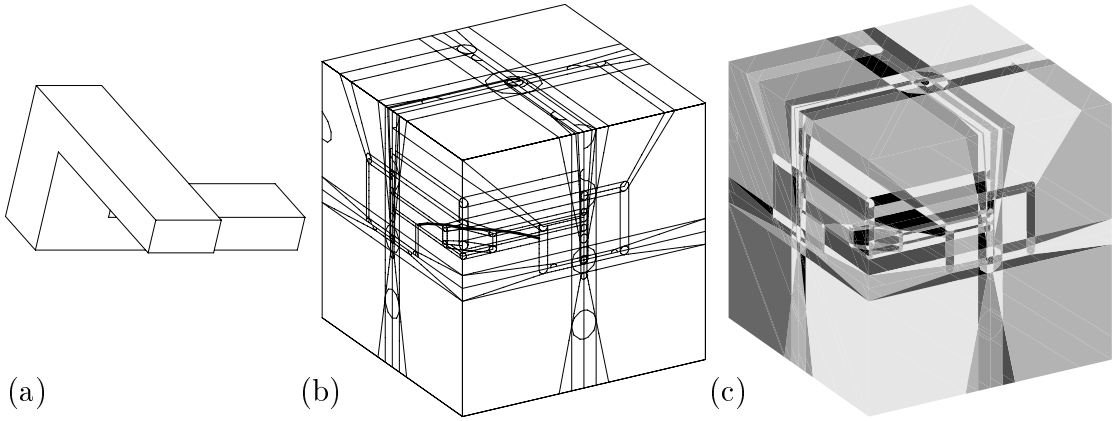


Figure 3.19 A "complex" object and its aspect graph: (a) the object; (b) the finite-resolution visual event curves; (c) the finite-resolution aspect graph.

One of the main problems with aspect graphs is that as the object gets more complex the aspect graph itself becomes unmanageably complex. At the resolution used in Figure 3.17, this problem is certainly not solved by our approach, since the finite-resolution aspect graph is larger than the infinite-resolution one. As ϵ grows larger, or object features get smaller, some of the aspect graph features disappear: Figure 3.20 shows the finite-resolution visual event curves of an object and the aspect graph for which some of the edges are shorter than ϵ. The aspect graph is considerably more simple than the previous one; in fact, the features due to the short edges appearing in the infinite-resolution aspect graph have disappeared. The aspect graph (which contains 74 regions) is shown in Figure 3.20(c). The aspect graph in Figure 3.20(c) is definitely simpler than the one shown in Figure 3.17(d).

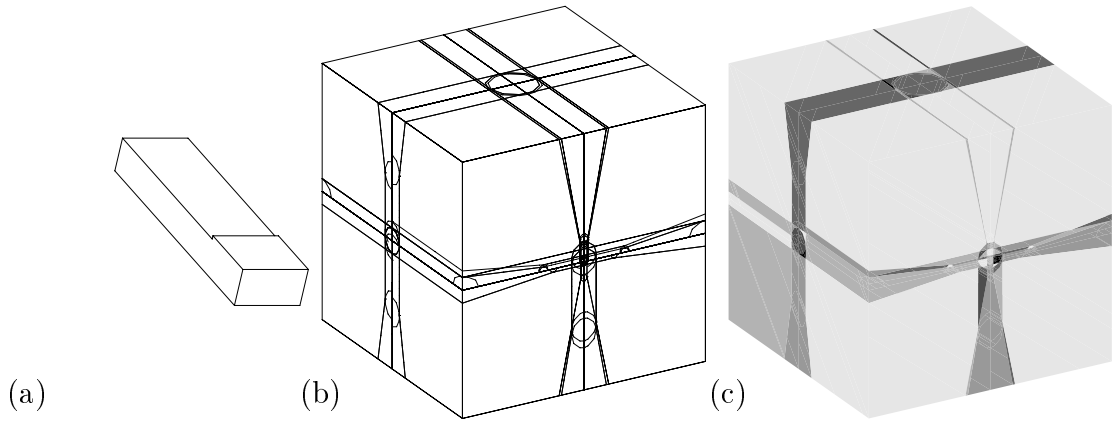


Figure 3.20 An object with edges shorter than ϵ and its aspect graph: (a) the object; (b) the finite-resolution visual event curves; (c) the finite-resolution aspect graph.

A more dramatic effect of the simplification power of the algorithm is shown in Figure 3.21. The object shown in Figure 3.21(a) consists of three connected orthogonal "legs". At the resolution at which we computed the finite-resolution aspect graph, the legs simplify to lines (Figure 3.21(b)). The infinite-resolution aspect graph shown in Figure 3.21(c) contains 120 regions. The finite-resolution visual event curves shown in Figure 3.21(d) bound 5060 regions, however after merging regions which have the same finite-resolution aspect, the number of regions is reduced to 20 (Figure 3.21(e)), which is considerably less then for the infinite-resolution aspect graph. The finite-resolution aspect graph is smaller than the infinite-resolution aspect graph in this case because the infinite-resolution aspect graph has to account for the different appearances of each leg and other small changes in appearance which do not exist in the finite-resolution aspects. A catalogue of all qualitatively different finite-resolution aspects is shown in Figure 3.22.

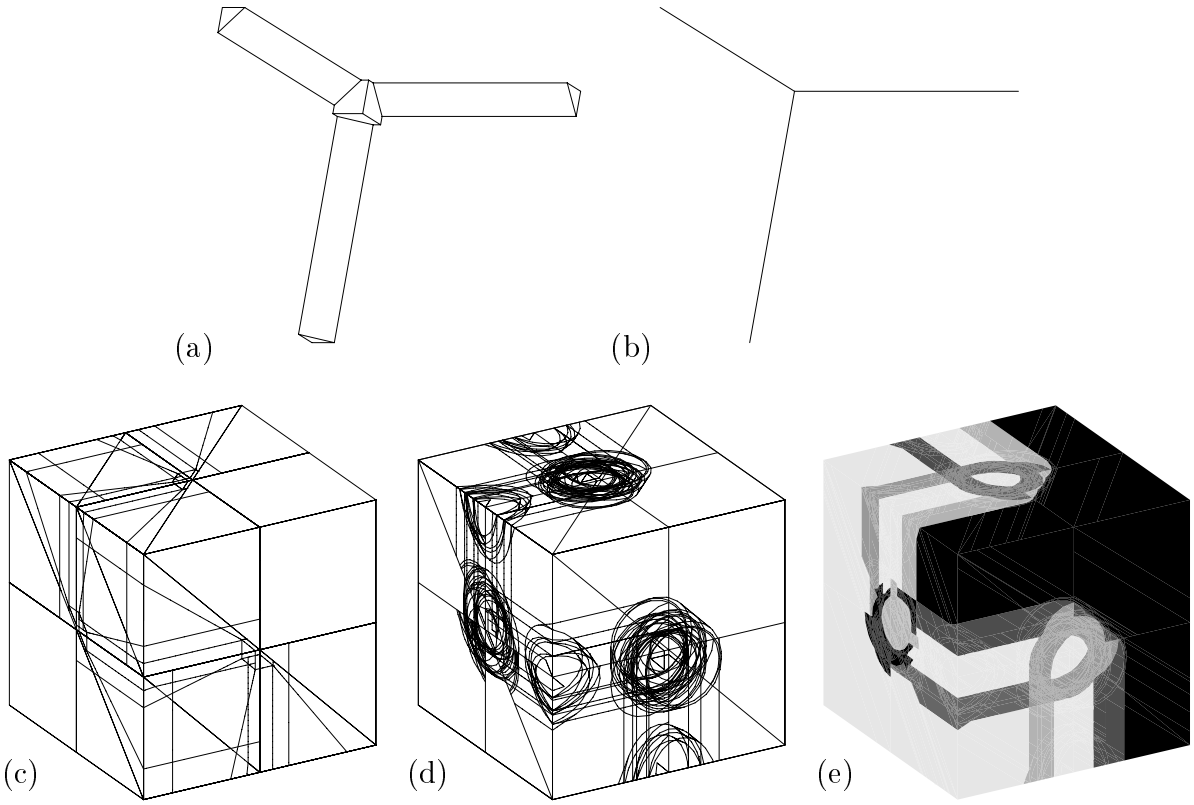


Figure 3.21 An object consisting of three orthogonal legs and its aspect graph: (a) the object; (b) its finite-resolution aspect; (c) the infinite-resolution visual event curves; (d) the finite-resolution visual event curves; (e) the finite-resolution aspect graph.

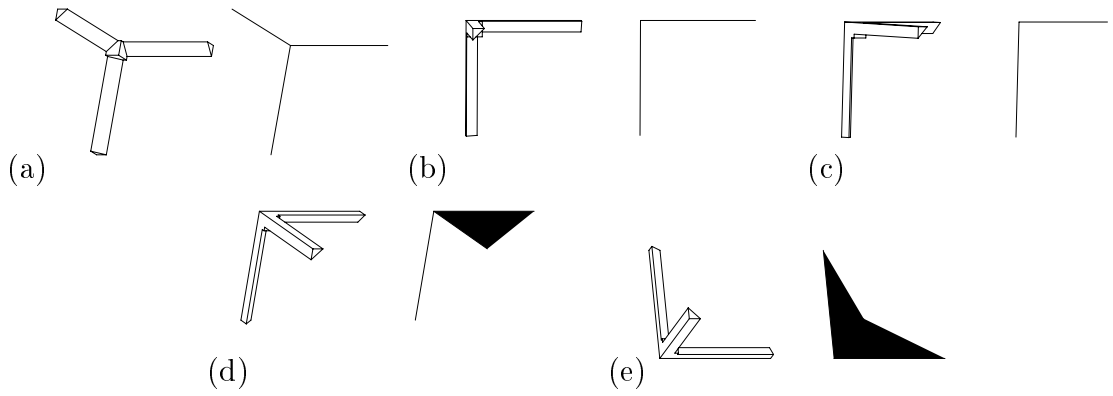


Figure 3.22 All qualitatively different finite-resolution aspects of an object consisting of three orthogonal legs: (a) all three legs are visible in the finite-resolution aspect; (b) one of the legs is foreshortened to a point in the finite-resolution aspect; (c) two legs appear as one line in the finite-resolution aspect; (d) two legs are close to each other but the width of the combined feature is more than ϵ and therefore they appear as a region in the finite-resolution aspect; (e) one of the legs is close to the other two legs but they are not close to each other causing all three of them to appear as a region in the finite-resolution aspect.

3.8 Discussion

We have attacked the problem of computing the exact aspect graph of a polyhedral object observed by an orthographic camera with finite-resolution, presented a catalogue of visual events for polyhedra observed under this projection model, and given an algorithm for computing the aspect graph and enumerating all qualitatively different aspects. Examples from the implementation of the algorithm have been presented.

One of the main problems with aspect graphs is that as the object gets more complex the aspect graph itself becomes unmanageably complex. An example of such an object and its infinite-resolution visual event curves is shown in Figure 3.23. The more features in the object that are visible from other features, the more complex the aspect graph becomes. The problem is even more acute when trying to generate finite-resolution event curves. In this example the cost was so prohibitive that we were not able to generate the finite-resolution visual event curves. So even though the final finite-resolution aspect graph can be quite simple as was shown in the example in Figure 3.21, the cost of

producing the finite-resolution aspect graph could be prohibitively high. Thus more work is required to find ways to prune the number of finite-resolution event curves to enable us to efficiently compute finite-resolution aspect graphs of more complex objects. More work is also required to compare the sizes of finite- and infinite-resolution aspect graphs as a function of resolution and object complexity.

Future work will be dedicated to actually using the finite-resolution aspect graph in recognition tasks, maybe in conjunction with our work on qualitative line-drawing analysis in the presence of uncertainty in vertex position (chapter 2). The main challenge will be to better predict and interpret merged image features such as those appearing in Figure 3.18.

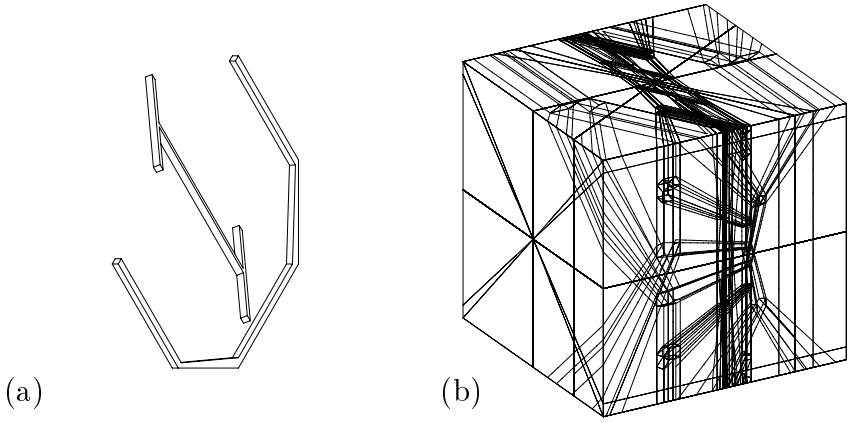


Figure 3.23 A very non-convex object and its infinite-resolution visual event curves: (a) the object; (b) the infinite-resolution visual event curves.

CHAPTER 4

Probabilistic 3D Object Recognition

4.1 Introduction

One of the major problems in computer vision is to recognize 3D objects appearing in a scene as instances from a database of models (see [12, 19] for reviews). Several recognition systems extract features such as points [46, 66, 67] or lines [7] from the image and match them to corresponding features in the database. They then verify each candidate hypothesis using other features in the image and finally rank the verified hypotheses.

When constructing such a system the major problems to be addressed are: how to generate match hypotheses, in which order to process the hypotheses, how to verify them using additional image features, and how to rank the verified hypotheses. In our approach, we define a match hypothesis as the matching of a pair of edges or a trihedral corner (Figure 4.1) to corresponding feature sets in the model database. The hypotheses are ranked by computing the probability that each one is correct. In order to verify a hypothesis, we compute the pose of the object assuming the match is correct. Other hypotheses which match other image features to features in the same model should yield compatible poses if they correspond to the same object. The pose is transformed from a

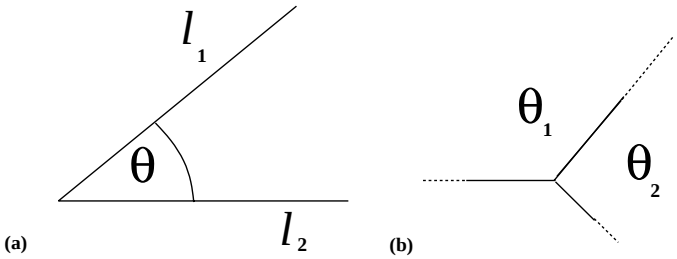


Figure 4.1 Image feature sets: (a)two lines in an image l_1 and l_2; the angle between the lines is θ; (b) a corner with three edges emanating from it.

point in the pose space to a region of that space when uncertainty in the values measured in the image is taken into account, and we check whether hypotheses reinforce each other by testing whether their pose uncertainty regions intersect. We rank sets of hypotheses whose uncertainty regions intersect by the probability that the match of the whole set is correct, and output a small number of interpretations with the highest ranks. The main goal of our approach is to find a small number of probable interpretations which can then be verified by comparing the image to the hypothesized object using standard verification techniques.

4.2 Literature Review

Since object recognition is one of the main problems in computer vision, a large body of work has addressed this problem (see [12, 19] for reviews). Here we focus on a few papers whose approach is related to ours.

Lowe [56] was the first to introduce the notion that a partial match of image to model features yields constraints on the position in the image of other features of the model due to rigidity constraints. Gaston and Lozano-Pérez [33] and Grimson and Lozano-Pérez [38] use the term *interpretation tree* for a general recognition paradigm: the idea is to order all possible matches between image and model features into a tree where each path represents an assignment of the image features to model features. The tree is searched for

consistent assignments. The size of the tree is exponential in the number of model and image features. Since searching the whole tree is prohibitively costly, they use geometric constraints (e.g., distance between points and angles between lines) to prune the tree from impossible combinations of matches. To further prune the search object recognition algorithms have been developed to limit the search by testing the more likely matches first by using rigidity constraints.

In [7], Ayache and Faugeras present a recognition algorithm for 2D objects which uses rigidity constraints. The algorithm extracts line segments from the image, and matches them to edges in the model. By assuming that a first match hypothesis is correct the algorithm constrains the pose of the object and can estimate the locations in the image of the projections of other edges. By adding to the hypothesis another match, the pose is completely determined. More matches reduce the pose uncertainty and help verify the hypothesis. In [28], Faugeras and Hebert present recognition algorithm for range images which also relies on rigidity constrains to select the next match and recover the pose of the object at the same time.

In [46], Huttenlocher and Ullman present a 3D object recognition algorithm based on the *alignment* technique. The algorithm extracts points from the image, and a match hypothesis is a match between a point triplet in the image and a point triplet in the model. Using this match, the pose of the object is recovered [1, 40]. Thus the position of other points in the image can be estimated and if found help verify the hypothesis. This algorithm is a special case of the *interpretation tree*, where the features matched are points and the search is terminated at the third level of the tree after three points have been matched. One of the most important problems that arises in [46] is in which order to test the $O(n^3m^3)$ match hypotheses, where n is the number of image points and m the number of model points because testing all of them is prohibitively costly.

Research on topics raised by this approach has been pursued in various directions. In [37], Grimson et. al. estimate the uncertainty in the pose of the object given a possible match of three points in the model to points in the image whose position is uncertain. In [2], Alter and Jacobs perform an analysis to find the position uncertainty of points in the image given a match of a number of points in the model to points in the image whose position is uncertain. The large uncertainty presented in that paper underlined the problem of relying on a small number of matching points to predict the position of other points in the image. In [36], Grimson and Huttenlocher perform a statistical analysis to determine what is the probability for a random match of image features to model features to occur, and they use this analysis to determine a threshold for the number of matched features to be used to recognize 2D objects.

Probabilistic methods have been developed to rank hypotheses by their likelihood in order to refrain from testing all possible match hypotheses. These methods are based on the probabilistic peaking effect which was first investigated by Arnold and Binford [3], and further studied by Binford Levitt and Mann [13], Ben-Arie [11], and Burns, Weiss, and Riseman [15]. Informally this effect can be described as follows: when randomly choosing a direction from which to observe a pair of adjacent edges, and measuring the ratio of the lengths of the projections of the edges or the angle between them in the image, the values measured in the image have a high probability to be close to the real values measured in the model. In those papers, probability density functions for values measured in the image are estimated by uniformly sampling the viewing sphere. In addition, joint density functions for ratios of lengths and angles are estimated. The maximum of those density functions are always where the measured values equal the real values measured in the model. Using Bayes's rule, these probabilities can be used to estimate the probability that a set of features in the image match a set of features in the model.

The probabilistic peaking effect is used by Olsen [66, 67] to rank match hypotheses, by Ben-Arie [11] to match line segments in the image to edges in the model database, and by Burns, Weiss, and Riseman [15] to recognize objects using view description networks, where the probabilities are used to search the description networks.

In a recent paper [93], Wheeler and Ikeuchi present a 3D object recognition algorithm from range images. The match hypotheses match regions in the image to faces in the model database and hypotheses reinforce each other if the hypothesized faces are supposed to appear in the image from the presumed viewing direction using the aspect graph of the object.

In our work, we will gain a better understanding of the probabilistic peaking effect, present exact techniques to compute the probabilities associated with it, new methods to compute the pose and the pose uncertainty regions due to vertex position uncertainty, and a probabilistic method for ranking possible interpretations of the image.

4.3 Approach

In this chapter, we propose a 3D object recognition algorithm whose input is a set of straight lines extracted from a single image by an edge detector. In order to build a robust recognition system we have to account for the fact that the input is not perfect: objects may occlude each other, certain edges may not belong to any instance of any model in our model database, and not all edges appearing in the image will be extracted by the edge detector. We explicitly take all these factors into account.

We group the lines extracted from the image into two types of feature sets: pairs of adjacent lines and trihedral corners (Figure 4.1). For line pairs, we measure the lengths of edges and the angle between them, and measure only the angles between the lines for trihedral corners. Therefore only pairs of edges whose visible part were fully

recovered by the edge detector will yield valid hypothesis where as for trihedral corners that is not required. We have chosen these two types of features because under the scaled orthographic camera model that we use, they contain the minimal amount of information required to recover the viewing direction, and because the number of image and model features is linear in the number of vertices in the image and in our model database, reducing the number of hypotheses that have to be tested compared to testing all matches of vertex triplets in the image to vertex triplets in the model database (as in [46]).

In order to speed up the recognition process we rank the match hypotheses using the probabilistic peaking effect, and test them in the corresponding order. When the pose of the object is randomly selected, the probabilistic peaking effect says that values measured in the image such as ratios of lengths or angles between edges have a high probability to be close to the real values measured in the model. For the feature sets we have chosen, we measure for the pair of edges the ratio ρ of the lengths of the two edges and the angle θ between them, and measure two angles θ_1, θ_2 for the trihedral corner. We could have computed at recognition time for all the hypotheses the probability that the match is correct, but in order to speed up the algorithm we partition the (ρ, θ) and (θ_1, θ_2) spaces into rectangles and precompute the average probability of hypotheses with measured values within each rectangle. Under the scaled orthographic camera model, the only component of the pose affecting the ratios and angles in the image is the viewing direction which is parameterized as a point on the viewing sphere. The area on the viewing sphere where the measured values are within a rectangle correspond to the required probability. This area is bounded by curves on the viewing sphere where the ratio (iso-ratio curve) or the angle (iso-angle curve) is constant, and equal to the minimum or maximum values of each rectangle. We can accurately compute the desired probabilities by computing the area bounded by these curves. We use the probabilities computed for all model

feature sets to compute the probability that a match hypothesis is correct given that we measured certain values in the image. We compute these probabilities using Bayes's rule and account in the probabilistic model for self occlusion and imperfect edge data. The model can be extended to incorporate more information about the image (e.g., the camera position and its relation to the surface on which the objects are placed), by reducing the space of legal poses to stable poses of the objects. This would reduce the number of plausible hypotheses for each feature set considerably.

We verify a match hypothesis by computing the pose of the object, assuming the hypothesis is correct. We then try to find consistent match hypotheses with close poses that reinforce the hypothesis. We compute the viewing direction component of the pose of the object by tracing the iso-ratio or iso-angle angle curves on the viewing sphere which correspond to the ratios and angles measured in the image, and the viewing direction is the intersection point of those curves. The other components of the pose are easily obtained. This technique works for adjacent edges and trihedral corners, where previous work developed separate techniques for them [1, 46, 95].

Uncertainty in image data causes uncertainty in the pose. So in order to be able to decide whether two hypotheses reinforce each other, we estimate a region of the pose space which accounts for that uncertainty, so that only hypotheses whose regions intersect reinforce each other.

In the final stage of the algorithm we use a probabilistic expression to rank match hypotheses. Analysis of this expression shows that it has the following characteristics: hypotheses which would cause large errors in the measurements in the image are ranked lower than hypotheses with smaller errors, and combinations of feature sets which appear often in the model database such as rectangular faces are ranked lower than more rare feature combinations.

The rest of the chapter is organized as follows. In Section 4.4 we develop a method for computing probability density functions using equations we develop for angles and ratios. We use these probabilities for ranking match hypotheses in Section 4.5. We discuss pose estimation and estimate the effects of uncertainty on the pose in Section 4.6. We present our probabilistic expression to rank match hypotheses in Section 4.7. We show experimental recognition results in Section 4.8. Finally, a number of issues raised by our work and future research directions are discussed in Section 4.9.

4.4 Hypothesis Probability Computation

In this section we compute the probability that a given match hypothesis is correct. We compute probabilities and probability density functions for some observable image quantity (e.g., the ratio between lengths or an angle) to have a given value when the value (measured in the model) is given. We then extend this technique to joint probabilities and probability density functions for two image quantities, and use the results to rank match hypotheses.

4.4.1 Iso-ratio and Iso-angle Curve equations

When measuring the ratio of line lengths in an image or the angle between lines, we want to know from what viewing directions this image could be scanned, given a match between the lines and certain edges in the model. Consider two segments l_1 and l_2 in the image (Figure 4.1(a)) which are projections of edges $\mathbf{u}_1$ and $\mathbf{u}_2$ respectively in the model, such that the ratio between the length of l_1 and l_2 is ρ and the angle between them is θ.

The ratio between the lengths of the projections of $\mathbf{u}_1$ and $\mathbf{u}_2$ is ρ for viewing directions $\mathbf{v}$ which satisfy:

$$|\mathbf{u}_1 \times \mathbf{v}| - \rho|\mathbf{u}_2 \times \mathbf{v}| = 0. \tag{4.1}$$

Squaring this equation yields a quadratic equation in $\mathbf{v}$.

Viewing directions which satisfy:

$$(\mathbf{u}_1 \times \mathbf{v}) \cdot (\mathbf{u}_2 \times \mathbf{v}) - \cos\theta |\mathbf{u}_1 \times \mathbf{v}||\mathbf{u}_2 \times \mathbf{v}| = 0, \tag{4.2}$$

yield an angle θ between the projections of $\mathbf{u}_1$ and $\mathbf{u}_2$. Squaring this equation yields a equation of degree four in $\mathbf{v}$. Note that because only $\cos^2\theta$ appears in the equation, it also accounts for the curves corresponding to $\pi - \theta, 2\pi - \theta$ and $(\pi - \theta)$. These must be identified and eliminated.

We trace these curves to gain a better understanding of the probabilistic peaking effect and use the area bounded by them to calculate probabilities. For tracing the curves we use an algorithm for tracing algebraic curves [55] which relies on homotopy continuation [61] to find all curve singularities and construct a discrete representation of the smooth branch curves (see Appendix B for details). We add the constraint $|\mathbf{v}|^2 = 1$ to the equations derived earlier to insure that the viewing directions lie on the viewing sphere. We decided to trace these curves on the viewing sphere and not on a cube as was done in Chapter 3, since tracing curves on the cube reduces the number of variables by one but requires to run the algorithm six times. As the degree of the equations here is less than in the previous chapter, the overhead for using the cube outweighs the gain of tracing curves of equations with less variables.

For reasons of symmetry, ratios are plotted using a $\log_2$ scale. In this way, the ratio and its inverse are symmetric around zero ($\log_2 \rho = -\log_2(1/\rho)$), and yield similar curves. We traced curves for values of ρ with equal intervals on the $\log_2 \rho$ scale. Figure 4.2(a) shows curves for values $\log_2 \rho$ between -2 and 2 where the the ratio in the model is 1. Interesting viewing directions are: viewing directions parallel to $\pm\mathbf{u}_1$ where $\mathbf{u}_1$ is foreshortened to a point and ρ is zero, viewing directions parallel to $\pm\mathbf{u}_2$ where $\mathbf{u}_2$ is foreshortened to a point and ρ is infinite and viewing directions parallel to $\pm(\mathbf{u}_1 \times \mathbf{u}_2)$ where the viewing direction is orthogonal to the plane of $\mathbf{u}_1$ and $\mathbf{u}_2$ and the ratio is the

"real" ratio. For ratios less than the "real" ratio there are two curves which surround the viewing direction parallel to $\pm\mathbf{u}_1$, for ratios greater than the "real" ratio there are two curves which surround the viewing direction parallel to $\pm\mathbf{u}_2$, and for the ratio equal to the "real" ratio the two curves intersect at the viewing directions parallel to $\pm(\mathbf{u}_1 \times \mathbf{u}_2)$.

For the case when the "real" ratio is not one, consider a viewing direction $\mathbf{v}$ which satisfies

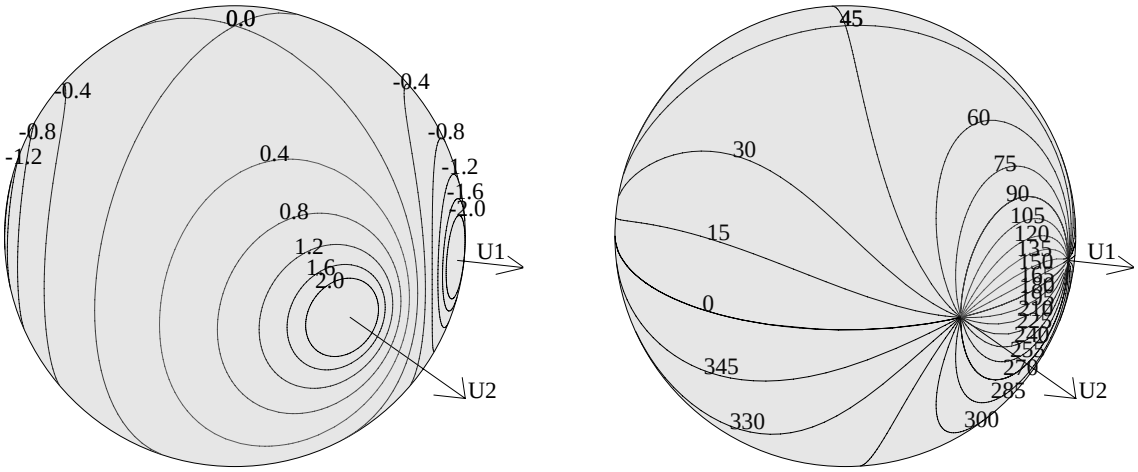
$$|\mathbf{u}_1 \times \mathbf{v}| - \rho|\mathbf{u}_2 \times \mathbf{v}| = 0 \tag{4.3}$$

for a pair of edges e_1 and e_2 such that $|\mathbf{u}_1| = |\mathbf{u}_2|$. Consider another pair of edges e_1' and e_2' such that $\mathbf{u}_1' = \lambda_1\mathbf{u}_1$ and $\mathbf{u}_2' = \lambda_2\mathbf{u}_2$. Substituting these values into (4.3) yields:

$$|\mathbf{u}_1' \times \mathbf{v}| - \rho\frac{\lambda_1}{\lambda_2}|\mathbf{u}_2' \times \mathbf{v}| = 0.$$

Thus the perceived ratio at the viewing direction $\mathbf{v}$ is $\rho\lambda_1/\lambda_2$ where λ_1/λ_2 is the "real" ratio between the lengths of e_1' and e_2'. This linear relationship enables us to concentrate on the case where the edges are of equal length and use this relationship to compute values for all other pairs of edges. No such relationship exists for angles.

In Figure 4.2(b) we show curves for angles between 0 and 360° where the angle is defined as the angle between the projection $\mathbf{u}_1$ the projection of $\mathbf{u}_2$ in the counterclockwise direction and the "real" angle is 45°. All the curves start at viewing directions parallel to $\pm\mathbf{u}_1$ and end at viewing directions parallel to $\pm\mathbf{u}_2$. This occurs because at those directions one of the edges has been foreshortened to zero length and therefore the angle between the two edges is not defined. However, by slightly changing the viewing direction any angle can be obtained. For angles less than the "real" angle, the curves go between $\mathbf{u}_1$ and $\mathbf{u}_2$ or between $-\mathbf{u}_1$ and $-\mathbf{u}_2$ starting for 0° from directions in the plane spanned by $\mathbf{u}_1$ and $\mathbf{u}_2$. At the "real" angle the two curves intersect like before at the viewing direction orthogonal to the $\mathbf{u}_1$,$\mathbf{u}_2$ plane. For angles greater than the "real" angle the curves go between $\mathbf{u}_1$ and $-\mathbf{u}_2$ or between $-\mathbf{u}_1$ and $\mathbf{u}_2$ ending for 180° with viewing



(a) (b)

Figure 4.2 For two lines of equal length with 45° between them: (a) the viewing sphere with iso-ratio curves with $\log_2(\rho)$ in $[-2, 2]$; (b) the viewing sphere with iso-angle curves.

directions which also lie in the $\mathbf{u}_1$,$\mathbf{u}_2$ plane. For angles θ greater than 180° the curve is the reflection of the curve for $360° - \theta$ through the origin. Because if the angle between the projections of $\mathbf{u}_1$ and $\mathbf{u}_2$ for viewing direction $\mathbf{v}$ is θ, the angle for $-\mathbf{v}$ is $360° - \theta$. The probabilistic peaking effect is clearly demonstrated in this figure for both ratios and angles. The fraction of the sphere covered by ratios such that $-0.4 < \log_2 \rho < 0.4$ and by $30° < \theta < 60°$ is much larger than by other segments of the $\log_2 \rho$ and θ spaces of the same size.

4.4.2 Computing Probability Density Functions

Given the probability density function (p.d.f.) f, the value of the distribution function $F(a)$ is the probability that $x \leq a$. It is given by:

$$F(a) = \int_{-\infty}^{a} f(x)dx.$$

Conversely,

$$f(x) = F'(x),$$

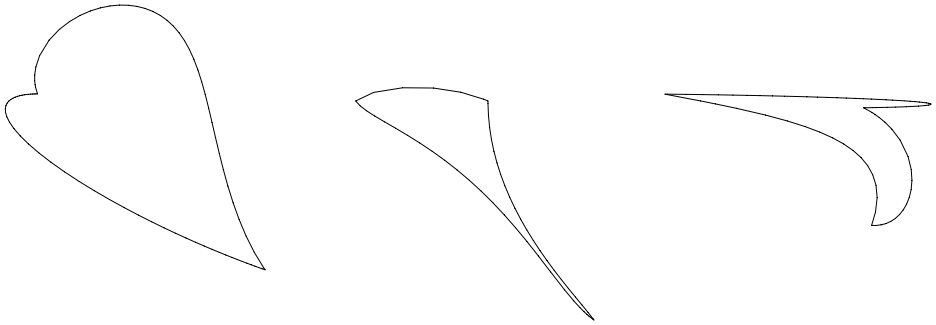


Figure 4.3 Regions on the viewing sphere whose areas are computed.

therefore when one function is given, the other can be computed by integration or differentiation. In our case we first compute $F(a)$ for a ratio or an angle and then compute f by numerical differentiation. The curve for a value of the ratio or an angle a is traced on the viewing sphere. This curve bounds the part of the viewing sphere for which $x \leq a$. The discrete points on the curve bound a polygon whose area is given by the following formula:

$$\text{area} = \sum_{i=1}^{n} \alpha_i - (n-2)\pi,$$

where n is the number of vertices of the polygon, and α_i is the spherical angle between two adjacent edges. A spherical angle is defined as the angle between the planes containing the great circles of the two edges. The area of the whole sphere is 4π, so to obtain probabilities the result must be divided by 4π. Examples of such curves for various types of distribution functions computed throughout this chapter are shown in Figure 4.3.

We can estimate $f(x)$ by:

$$f(x) = \lim_{\epsilon \to 0} \frac{F(x+\epsilon) - F(x-\epsilon)}{2\epsilon},$$

using the values computed for $F(x+\epsilon)$ and $F(x-\epsilon)$.

Figure 4.4(a) shows the probability density function for $\log_2 \rho$ for various values of the original angle between the edges, where the original ratio is 1. A logarithmic scale is used to reflect the symmetry of the probability density function for ratios.

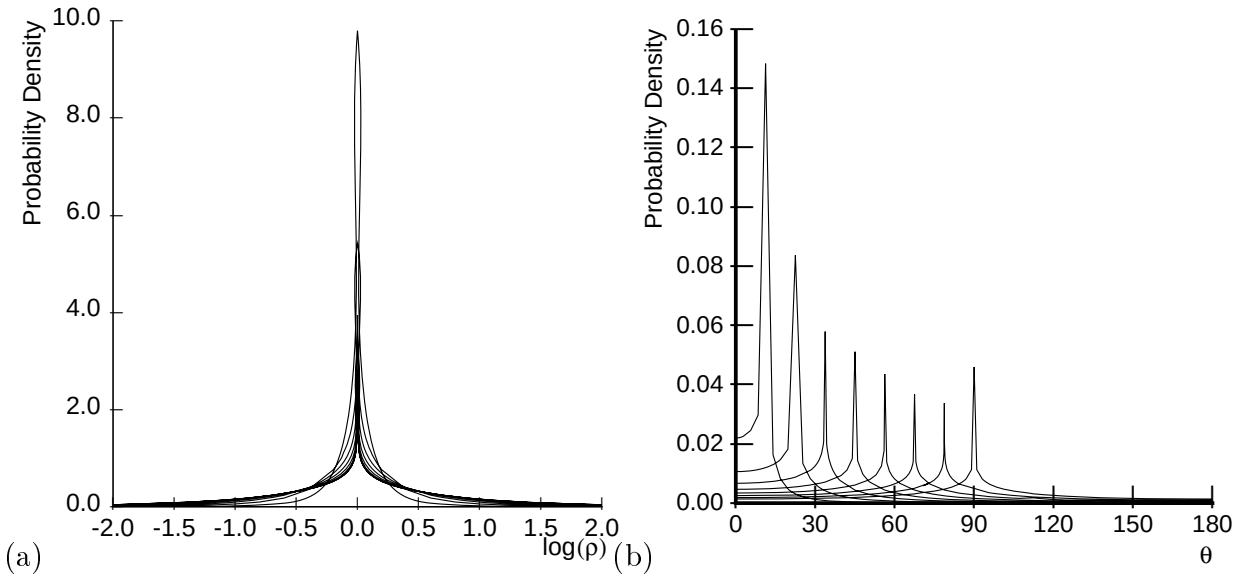


Figure 4.4 Probability densities for ratios and angles for various values of θ: (a) probability density for ratios; (b) probability density for angles.

Figure 4.4(b) shows the probability density function for the observed angle for various values of the original angle. In previous work Binford et. al. [13], Ben-Arie [11] and Burns et. al. [15] estimated probability density functions using ratios and angles computed of a uniform sample of the viewing sphere. However our method, which computes the areas on the viewing sphere produces the probability density functions exactly (up to the discretization error of the curves and the error due to finite differences).

4.4.3 Computing Joint Probability Density Functions

In the previous section we showed how to compute probability density functions for one quantity measured in the image. However, when performing a partial match of image features to model features such as the ones in Figure 4.1, more than one value is measured. Computing their joint probability density function can be used to rank the different match hypotheses for these image features. To demonstrate this we will examine the ratio-angle pair. The p.d.f. $f(\rho, \theta)$ and the distribution function $F(\rho, \theta)$ are related

in the following ways:

$$F(\rho,\theta) = \int_0^{\rho}\int_0^{\theta} f(x,\alpha)dxd\alpha,$$
$$f(\rho,\theta) = \frac{\partial F}{\partial\rho\partial\theta}(\rho,\theta),$$

where $F(\rho,\theta)$ is the probability that the ratio between the lines in the image is less than ρ and the angle between them is less than θ. We estimate $f(\rho,\theta)$ as:

$$f(\rho,\theta) = \lim_{\substack{\mu\to 0\\ \nu\to 0}} \frac{F(\rho+\mu,\theta+\nu)+F(\rho-\mu,\theta-\nu)-F(\rho+\mu,\theta-\nu)-F(\rho-\mu,\theta+\nu)}{2\mu 2\nu}, \tag{4.4}$$

where the numerator is the area bounded by the iso-ratio curves for $\rho\pm\mu$ and the iso-angle curves for $\theta\pm\nu$. The area is found by tracing the four curves, finding the intersection points between them using homotopy continuation, extracting the boundary of the region and computing its area.

4.4.4 Computing Joint Distribution Functions

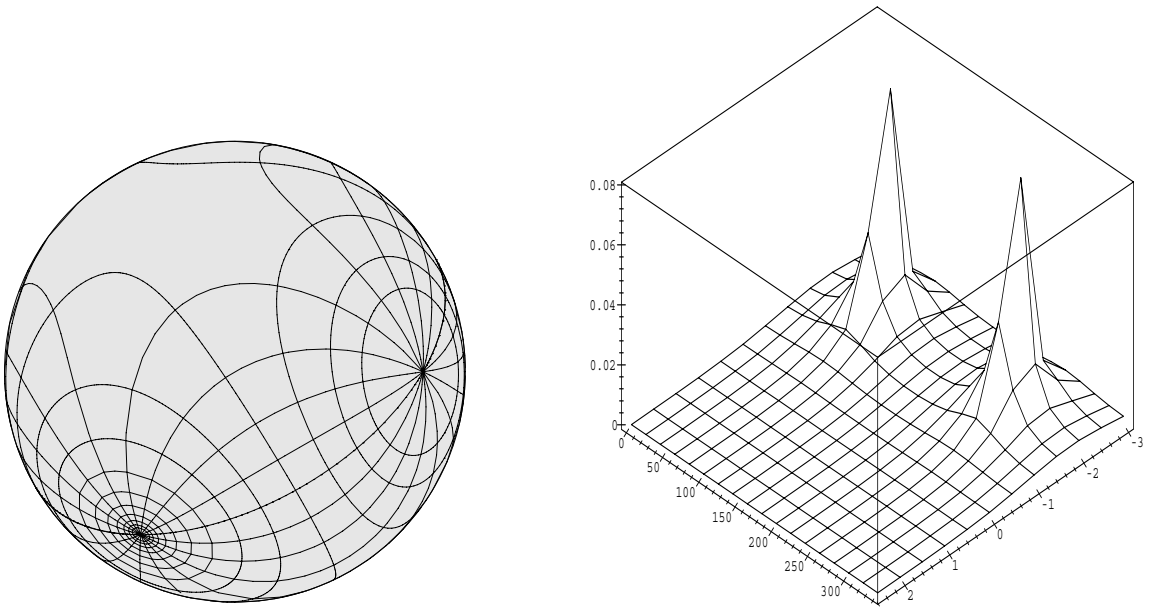
In order to rank match hypotheses we must be able to compute how likely the values measured in the image are when the match hypothesis is correct. The likelihood of a match hypothesis is measured by the value of the joint p.d.f. which was computed in the previous section for the values measured in the image. This can be done during the recognition process for all the match hypotheses. However in order to speed up the recognition process we build look-up tables off-line. The $(\log_2\rho,\theta)$ and (θ_1,θ_2) spaces are divided into rectangles and for each rectangle the average value of the joint p.d.f. is computed. At recognition time the value of the joint p.d.f. is found in the appropriate entry of the look-up table.

To perform that we have to compute for two adjacent edges e_1 and e_2 the probability

$$P(\rho_1<\rho<\rho_2,\theta_1<\theta<\theta_2), \tag{4.5}$$

that when viewing these edges from a randomly selected viewing direction the ratio of the lengths ρ will be between ρ_1 and ρ_2 and the angle between the edges will be between θ_1 and θ_2. We will denote this region of the (ρ, θ) space by $R(\rho_1, \rho_2, \theta_1, \theta_2)$. Using the technique described in the previous section we trace the curves of ρ_1, ρ_2, θ_1 and θ_2 and find the area bounded by them. Because for every value of ρ or θ, there are two curves, two such areas exist. We have divided the $(\log_2 \rho, \theta)$ space into identical rectangles. The $\log_2 \rho$ dimension was truncated at $-k$ and k where k is an arbitrary limit set taking into account the maximum and minimum lengths of edges assumed to be extracted by the edge detector, and using those values to estimate the maximum and minimum ratios of lengths. Dividing $P(\rho_1 < \rho < \rho_2, \theta_1 < \theta < \theta_2)$ by the area of the rectangle in the $(\log_2 \rho, \theta)$ space yields the average joint p.d.f. value for ratios and angles within that rectangle. We have traced the corresponding regions on the viewing sphere of each rectangle, producing a tessellation of the viewing sphere shown in Fig 4.5(a). The areas of the regions which represent $P((\rho, \theta) \in R(\rho_1, \rho_2, \theta_1, \theta_2))$ are plotted in Figure 4.5(b).

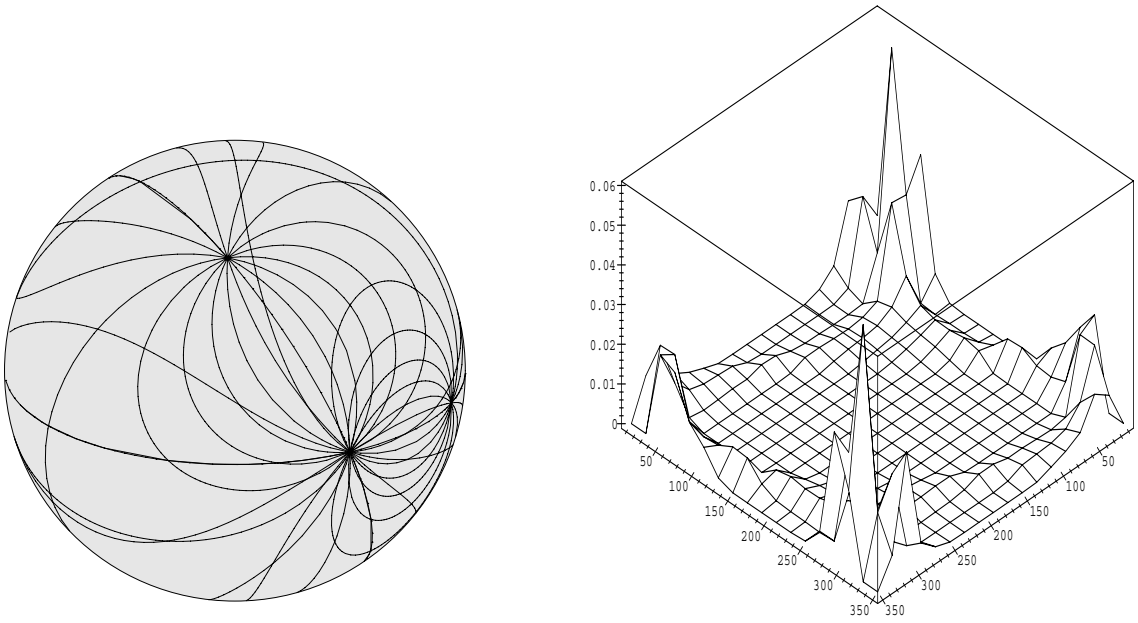
For trihedral corner feature sets curves for the two angles are plotted producing a tessellation of the viewing sphere shown in Figure 4.6(a). The areas of the regions which represent $P((\theta_1, \theta_2) \in R(\theta_{11}, \theta_{12}, \theta_{21}, \theta_{22}))$ were plotted in Figure 4.6(b). Corners are characterized by the three angles between the three pairs of edges. Although the two sets of curves (such as the set in Figure 4.2(b)) are determined by the first two angles, the third angle influences where the two sets of curves will intersect and thus the area of the regions (if they exist). To demonstrate how different the results can be, we have computed the joint probability function of a number of corners, and display in Figure 4.7(top) the graph of the probability function. The bottom of the figure shows the empty and non-empty regions. This shows that for a given corner there doesn't always exist a viewing direction for every pair of angles, where as in the ratio/angle case there exist viewing directions for all values of ρ and θ. This enables us to discard some of the



(a) (b)

Figure 4.5 The joint probability function of ratios and angles for a pair of edges where the original values are $(1/3, 90°)$: (a) tessellation of the viewing sphere into ratio/angle regions; (b) a graph of the joint probability function for ratios and angles as estimated by the area of the regions in (a).

hypotheses for trihedral corners because there is no viewing direction which would yield those angles for some of the model feature sets.



(a) (b)

Figure 4.6 (a) Tessellation of the viewing sphere into angle/angle regions; (b) a graph of the joint probability function for pairs of angles as estimated by the area of the regions in (a).

In the ratio/ratio case the feature set includes three edges. We could then measure in the image in addition to the two ratios, two angles. This means that the ratio/ratio case can be modelled as an angle/angle and two ratio/angle feature sets. So when a ratio/ratio feature set occurs in the image it is modelled as a combination of its simpler components.

4.4.5 Dealing with Occlusion

Up until now our analysis did not account for the fact that the features we are studying belong to solid objects and these objects could partially or totally occlude these features. The features can be occluded by features of the same object or by other objects. Modeling

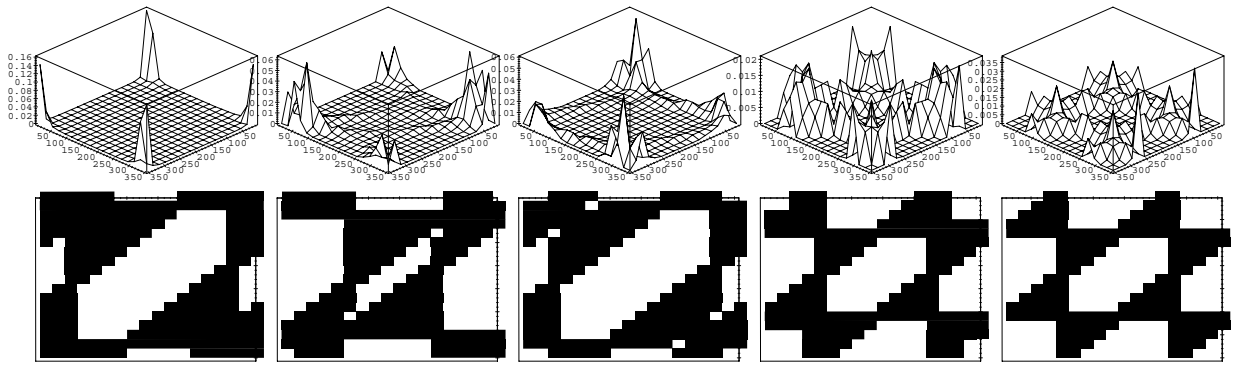


Figure 4.7 For trihedral corners with the following angles: $(10°, 20°, 25°)$, $(40°, 60°, 40°)$, $(40°, 60°, 80°)$, $(80°, 80°, 80°)$ and $(90°, 90°, 90°)$ graphs of joint probability functions and of non-empty regions are displayed.

the effects of occlusion between different objects is very difficult without having prior knowledge about the location of the objects and the camera in the scene.

Self occlusion on the other hand can be modelled exactly by dealing with the following two problems: for both types of feature sets the vertex at which the edges meet (the corner) must be visible, and when computing the ratio of lengths in two edge feature sets, only the length of the visible part of the projection of the edge should be used. We determine the visibility of the features using the aspect graph of the object which is generated using techniques similar to the ones described in chapter 3. As the visibility of features only changes on the viewing sphere on critical curves, we can analyze the visibility of the features in the feature set by studying a representative viewing direction from each noncritical region and determine where the features are visible and which critical curves bound those regions. Regions in which the corner is not visible are removed from the probability distribution computed earlier, and for areas in which one or both the edges are partially occluded we replace the curves traced for the ratio of lengths by new curves using the following derivation.

Given two edges e_1 and e_2 shown in Figure 4.8, we characterize the viewing directions $\mathbf{v}$ such that the ratio of the lengths of the visible part of their projections is ρ. So if the

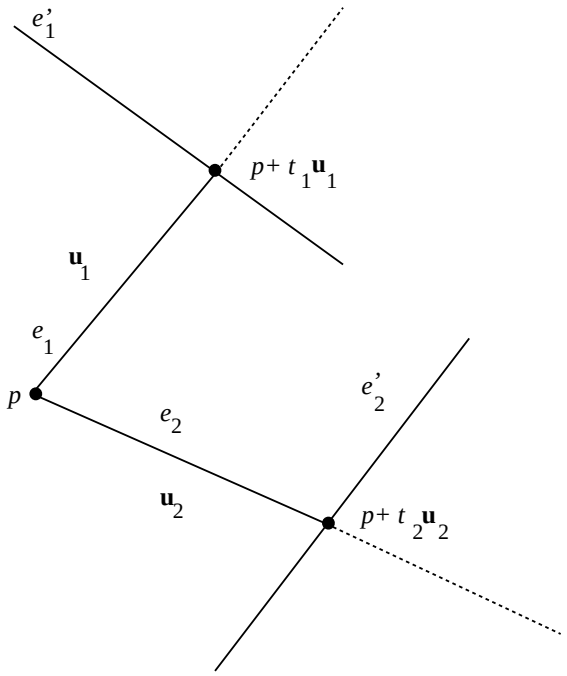


Figure 4.8 Two adjacent edges e_1 and e_2, starting from vertex p in directions $\mathbf{u}_1$ and $\mathbf{u}_2$ respectively, intersect edges e_1' and e_2' at $p + t_1\mathbf{u}_1$ and $p + t_2\mathbf{u}_2$ respectively.

projections of e_1 and e_2 intersect the projections of e_1' and e_2' at $p + t_1\mathbf{u}_1$ and $p + t_2\mathbf{u}_2$ respectively, (4.1) is modified into:

$$t_1|\mathbf{u}_1 \times \mathbf{v}| - \rho t_2|\mathbf{u}_2 \times \mathbf{v}| = 0. \tag{4.6}$$

In order to characterize the intersection points between the projections of the edges we use the same arguments used to derive the equations for EEE curves in Chapter 3. We write that a line $D_1^{''}$ passing through $p + t\mathbf{u}_1$ with direction $\mathbf{v}$ intersects the supporting line $D_1' = (a_1, b_1)$ of e_1'. The line $D_1^{''}$ has Plücker coordinates $(\mathbf{v}, (p + t_1\mathbf{u}_1) \times \mathbf{v})$. Writing that $D_1^{'}$ intersects $D_1^{''}$ we obtain:

$$a_1 \cdot (t_1\mathbf{u}_1 \times \mathbf{v} + p \times \mathbf{v}) + b_1 \cdot \mathbf{v} = 0.$$

Similarly we obtain for the intersection of e_2 and e_2':

$$a_2 \cdot (t_2\mathbf{u}_2 \times \mathbf{v} + p \times \mathbf{v}) + b_2 \cdot \mathbf{v} = 0.$$

Substituting the equations for t_1 and t_2 obtained from these equations into (4.6) yields:

$$\frac{-b_1 \cdot \mathbf{v} - a_1 \cdot (p \times \mathbf{v})}{a_1 \cdot (\mathbf{u} \times \mathbf{v})} |\mathbf{u}_1 \times \mathbf{v}| - \rho \frac{-b_2 \cdot \mathbf{v} - a_2 \cdot (p \times \mathbf{v})}{a_2 \cdot (\mathbf{u} \times \mathbf{v})} |\mathbf{u}_2 \times \mathbf{v}| = 0. \qquad (4.7)$$

Squaring this equation yields a degree six homogeneous equation in $\mathbf{v}$. If only one of the edges is partially occluded the equation simplifies to an equation of degree four.

To demonstrate the effect of occlusion we have chosen two edges of an L shaped object whose aspect graph is shown in Figure 4.9(a). We labeled the noncritical regions of the aspect graph with letters $A - F$ according to the state of occlusion of the two edges and show representative views of the object in Figure 4.9(b). In regions labeled by A both edges are totally visible therefore (4.1) is used to trace the iso-ratio curves. In regions labeled by B at least one of the edges is totally occluded, and therefore these regions are discarded. In C one of the edges is partially occluded and in D and E the other edge is partially occluded, each time by a different edge. For these regions (4.7) is used to trace the iso-ratio curves. In region F the corner is occluded therefore this region is also discarded.

In Figure 4.9(c) the aspect graph and the tessellation of the sphere into regions are traced on the viewing sphere. It is interesting to note the effect of crossing a critical curve on the iso-ratio curve. On the boundary between regions A and E the iso-ratio curve is continuous where as on the boundary between regions A and D or A and C it is not. In the former when the viewing direction is part of the critical curve the projection of the occluding edge touches a vertex of one of the edges in the feature set and therefore does not occlude it at all. In the latter when the critical curve is passed the edge changes from being totally visible to being partially occluded changing the ratio of the visible parts of the projections of the edges instantaneously. In region F the corner is occluded and on its boundaries e_1 (C) and e_2 (D and E) become totally occluded yielding sets of dense iso-ratio curves with values converging on zero and infinity respectively. The probability distribution function is shown in Figure 4.9(d) (compare to Figure 4.5 which

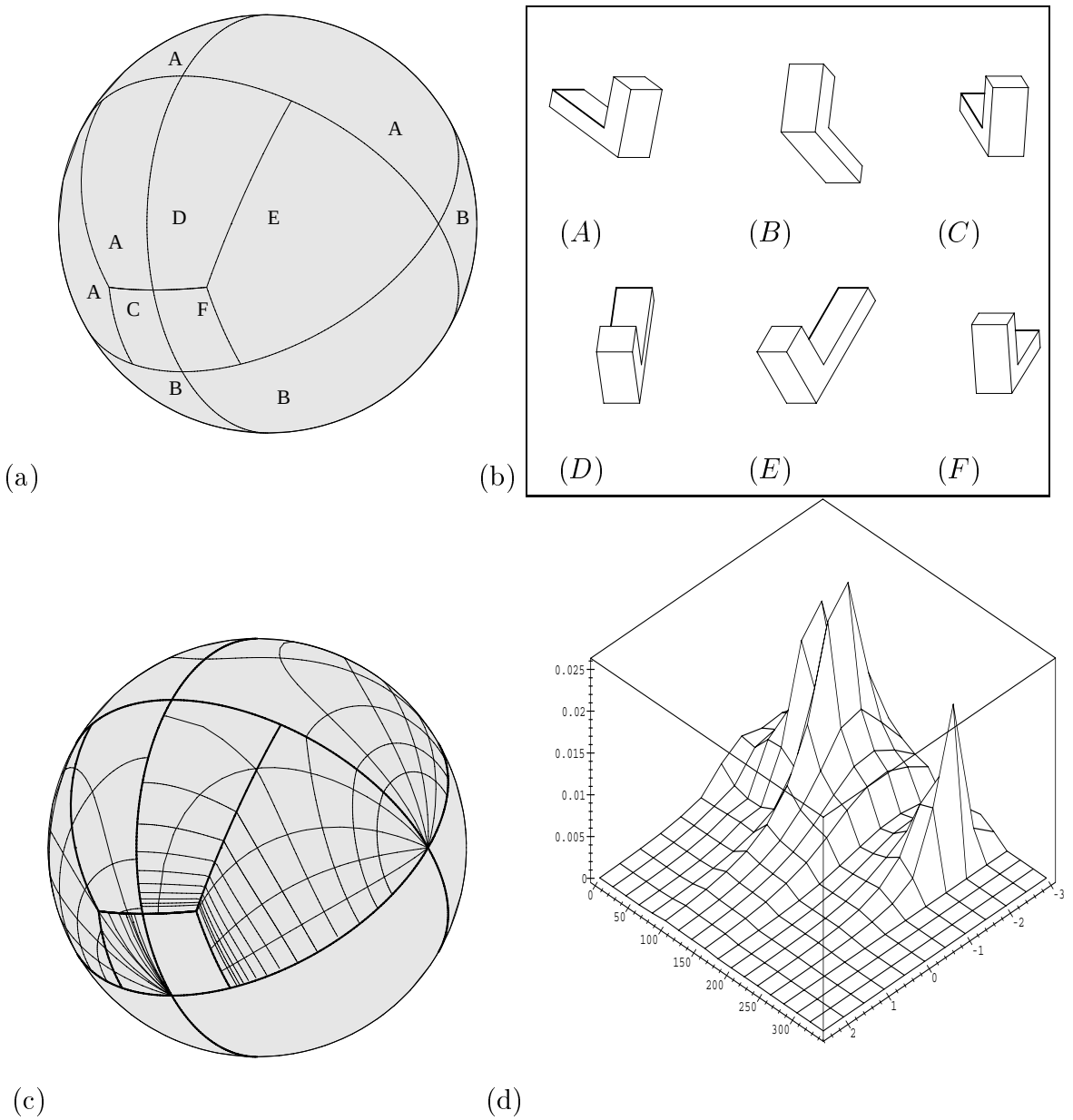


Figure 4.9 (a) the aspect graph of the L shaped object with the regions marked by letters A-F; (b) views of the object: (A) both edges are visible; (B) at least one of the edges is not visible; (C) one of the edges is partially occluded; (D)-(E) the other edge is partially occluded by two different edges; (F) the corner is occluded; (c) tessellation of the viewing sphere into regions; (d) a graph of the joint probability function as estimated by the area of the regions in (c).

disregards occlusion). Similar probability distributions have been computed for the case of the trihedral corner. In that case regions on the viewing sphere where the corner is not visible were discarded. In the other regions the regular iso-angle equations were used.

4.5 Ranking Match Hypotheses

In this section we use the techniques presented in the previous section for ranking match hypotheses. We build look-up tables of lists of hypotheses sorted by probability off-line and during the recognition stage the sorted hypotheses are tested.

4.5.1 Building Look-up Tables

In the preprocessing stage, we construct two look-up tables $T_1(\rho, \theta)$ and $T_2(\theta_1, \theta_2)$ for the ratio/angle and angle/angle pairs respectively. The tables are built by computing joint probability distributions which were described in Section 4.4.4 for all feature sets in the model database.

We use these probabilities to determine the induced probabilities on the identity of the object. We denote by E a feature set measured in the image with values within a certain region of the (ρ, θ) or (θ_1, θ_2) space, by M_i the event in which the i^{th} model from the database appears in the image, and by $H_j^{(i)}$ the hypothesis that the j^{th} model feature set of the i^{th} model matches E. We would like to compute $P(H_j^{(i)}, M_i|E)$ which is the probability that hypothesis $H_j^{(i)}$ is correct if E was measured in the image. Using Bayes' law

$$P(H_j^{(i)}, M_i|E) = \frac{P(E, H_j^{(i)}|M_i)P(M_i)}{P(E)}.$$

$P(E, H_j^{(i)}, |M_i)$ is the probability that features associated with hypothesis $H_j^{(i)}$ were visible and that the values measured in the image were within the region of E assuming that object M_i is visible in the scene. This probability was computed in the previous

section. $P(E)$ is the sum of all the probabilities of all the hypotheses in which E can be measured. Therefore,

$$P(E) = \sum_m \sum_j P(E, H_j^{(m)}|M_m)P(M_m).$$

Combining these two equations yields:

$$P(H_j^{(i)}, M_i|E) = \frac{P(E, H_j^{(i)}|M_i)P(M_i)}{\sum_m \sum_j P(E, H_j^{(m)}|M_m)P(M_m)}. \tag{4.8}$$

$P(M_i)$ can be determined using any prior knowledge we have about the likelihood of a model to appear in the image. $P(H_j^{(i)}, M_i|E)$ is computed for the hypotheses in the list for each entry in the tables and the lists are sorted by probability.

It is important to note that the regions in the (ρ, θ) or (θ_1, θ_2) spaces do not have to be of equal size. Moreover, the recognition algorithm will perform better if regions with multiple probabilistic peaks are divided into smaller regions with one peak in each. Thus each hypothesis would have a higher probability in the subregion in which its peak is located and a lower probability in the other subregion, where as if only one region existed all the hypotheses would have the same average probability. In general, the (ρ, θ) or (θ_1, θ_2) space can be partitioned using a quadtree approach where a region is divided into four regions only if the order of the hypotheses of the subregions is different than the order of the hypotheses of the parent region.

4.5.2 Dealing With Uncertainty

The input to the recognition phase of the algorithm is the result of edge and corner detection performed on the image. There is uncertainty in this data which has to be modeled in order to design a robust algorithm. We identify three types of uncertainty which have to be dealt with: certain feature sets which appear in the image will not be recovered by the edge and corner detectors, the image features might not match any model features, and the values measured for image features are themselves uncertain.

We denote by $P_d(E,H)$ the probability that a feature set which appears in the image, for which we measure values E, and for which the hypothesis H is correct will be detected by the edge and corner detectors. For each hypothesis evidence pair $P_d(E,H)$ can be in principle obtained empirically by testing the performance of edge and corner detectors on many typical scenes. In general, feature sets appearing on the silhouette of the object will have a higher $P_d(E,H)$ than internal features.

Incorporating $P_d(E,H)$ into (4.8) yields:

$$P(H_j^{(i)}, M_i|E) = \frac{P(E, H_j^{(i)}|M_i)P(M_i)P_d(E, H_j^{(i)})}{\sum_m \sum_l P(E, H_l^{(m)}|M_m)P(M_m)P_d(E, H_l^{(m)})}.$$

The second type of uncertainty deals with feature sets which are not "legal" feature sets. Image features may not match model features for various reasons: the features were not correctly recovered (e.g. only part of an edge was recovered), the features belong to several occluding objects, or the features belong to objects not in the database or to the background. The average number of "illegal" feature sets P_{il} in a scene depends on the types of scenes which the recognition process analyzes and on the quality of the edge and corner detectors. However, P_{il} can be estimated empirically by testing the system on images of typical scenes. Different types of environments will produce different values for P_{il}. In order to incorporate this information into $T_1(\rho,\theta)$ and $T_2(\theta_1,\theta_2)$ tables we compute the probability that we measured values within a region in the value space due to random features. Assuming that the length of the lines and the angles between them are uniformly distributed, the distribution of $\log_2 \rho$ which is the logarithm of the ratio of the lengths of two lines with uniformly distributed lengths, is $\frac{1}{2}e^{-|\log_2 \rho \ln 2|}$. The probability for a pair of edges to be in a rectangle in the (ρ,θ) space is:

$$P((\rho,\theta) \in R(\rho_1,\rho_2,\theta_1,\theta_2)) = \frac{1}{2}|e^{-|\log_2 \rho_1 \ln 2|} - e^{-|\log_2 \rho_2 \ln 2|}|\frac{(\theta_2-\theta_1)}{2\pi}.$$

The probability for a corner to be in a rectangle in the (θ_1,θ_2) space is:

$$P((\theta_1,\theta_2) \in R(\theta_{11},\theta_{12},\theta_{21},\theta_{22})) = \frac{(\theta_{12}-\theta_{11})}{2\pi}\frac{(\theta_{22}-\theta_{21})}{2\pi}.$$

In each entry in the tables a new hypothesis is added to the list, which represents the "illegal" feature set and whose probability is the probability calculated above multiplied by P_{il}.

The last type of uncertainty is in the values measured in the image. For the two-edge case we assume that there is uncertainty in vertex position of the three vertices. We assume the six coordinates $\mathbf{a} = (x_1, y_1, x_2, y_2, x_3, y_3)$ have a normal distribution with mean $\mu_i = a_i$ and variance σ^2 which can be determined empirically. This induces a probability distribution on ρ and θ which enables us to compute the probability that the actual measured values are in a region of the (ρ, θ) space. Thus

$$P(H|E) = \sum_k P(H|E_k)P(E_k|E) = \sum_k P(H|E_k)\frac{P(E|E_k)P(E_k)}{\sum_j P(E|E_j)P(E_j)},$$

where $P(E|E_k)$ is the probability that E is in the k^{th} region of the (ρ, θ) space. $P(E|E_k)$ decreases rapidly as the distance from E to the region E_k increases, and the rate of the decrease accelerates as σ^2 gets smaller. Therefore for most regions E_k except the ones close to E, $P(E|E_k)$ is negligible and $P(H|E)$ is be computed by summing over the contributions of a small number of regions.

4.6 Pose and Pose Uncertainty Estimation

For all correct hypotheses which match features from an instance of an object to features in the model, the pose recovered for all of these match hypotheses should be the same. Therefore we use pose estimation to find sets of hypotheses which produce the same pose to reinforce the recognition hypothesis. Uncertainty in the values measured in the image induces uncertainty in the pose and are accounted for when testing the compatibility of match hypotheses.

4.6.1 Pose Estimation

The problem of estimating the pose from three points [1, 40, 46] or a trihedral corner [95] has been extensively studied. We present here a simple approach which deals with both types of feature sets. We regard the pose of an object in weak perspective projection as a combination of the following four components:

- a viewing direction $\mathbf{v}$ which is a point on the viewing sphere,
- a rotation of the image by δ degrees about the viewing direction,
- a scale s,
- a translation $\mathbf{t}$ in the image.

Therefore the pose is a point in the six dimensional space $\{S^2 \times [0 \cdots 2\pi] \times [s_{min} \cdots s_{max}] \times \mathbb{R}^2\}$ where S^2 is the unit viewing sphere, s_{min} and s_{max} are the minimum and maximum assumed scales respectively, and the translation $\mathbf{t}$ must be a vector in the image. The projection p_i of a point p of the object in the image is:

$$p_i = sR(\delta)(p \cdot \mathbf{v}_2, p \cdot \mathbf{v}_3) + \mathbf{t}, \tag{4.9}$$

where $\mathbf{v}, \mathbf{v}_2, \mathbf{v}_3$ is an orthonormal basis of $\mathbb{R}^3$ and $R(\delta)$ is a rotation by δ degrees.

We will now show what components of the pose can be recovered from angles and lengths measured in the image given an hypothesis which matches a minimal number of features in the image to features in the model. Each measured angle or ratio imposes a one-dimensional constraint on possible viewing directions. In order to determine $\mathbf{v}$ two such constraints are needed. Two pairs of types of curves were considered: the ratio/angle pair and the angle/angle pair. The ratio/ratio pair is not considered because in order to measure two ratios we would need an hypothesis which contains additional matched features which we are trying to avoid. The ratio/angle pair occurs when two adjacent

edges in the image (Figure 4.1(a)) are matched with edges in the model. The angle/angle pair occurs when a trihedral corner (Figure 4.1(b)) is matched to corresponding features in the model. In this case only the angles are used and the length of the edges does not have to be reliably measured in the image. $\mathbf{v}$ is obtained as the intersection points between two curves. The degrees of the ratio and angle curves are 2 and 4 respectively. Therefore the number of intersection points found for the ratio/angle and angle/angle pairs will be at most 8 and 16 respectively. However as shown in Section 4.4 the angle equations generate curves for $\theta, \pi-\theta, 2\pi-\theta$ and $-(\pi-\theta)$. As most of the solutions are for the other angles the number of real solutions is much less. Some of the remaining solutions can be eliminated by visibility considerations (i.e., the features are occluded from that viewing direction). The other components of the pose are determined using standard 2D pose estimation. The rotation angle δ is determined for both pairs by rotating the results of applying the projection of the object in direction $\mathbf{v}$ (Figure 4.10(a)) until the corresponding edges are parallel to each other (Figure 4.10(b)). The scale and translation can not be obtained for the angle/angle case. Therefore only in the case of the ratio/angle pair they are recovered (Figure 4.10(c) and Figure 4.10(d)).

4.6.2 Pose Uncertainty Estimation

For each match hypothesis we are able to recover certain components of the pose of the object as described in the previous section. However to reinforce these hypotheses we need to find sets of hypotheses matching different feature sets in the image to feature sets of the same model in the database and determine that these features belong to the same instance of the model. In that case the values of the components of the pose recovered for each match hypothesis should be the same. When uncertainty in the values measured in the image is accounted for, instead of points in the pose space we get pose uncertainty

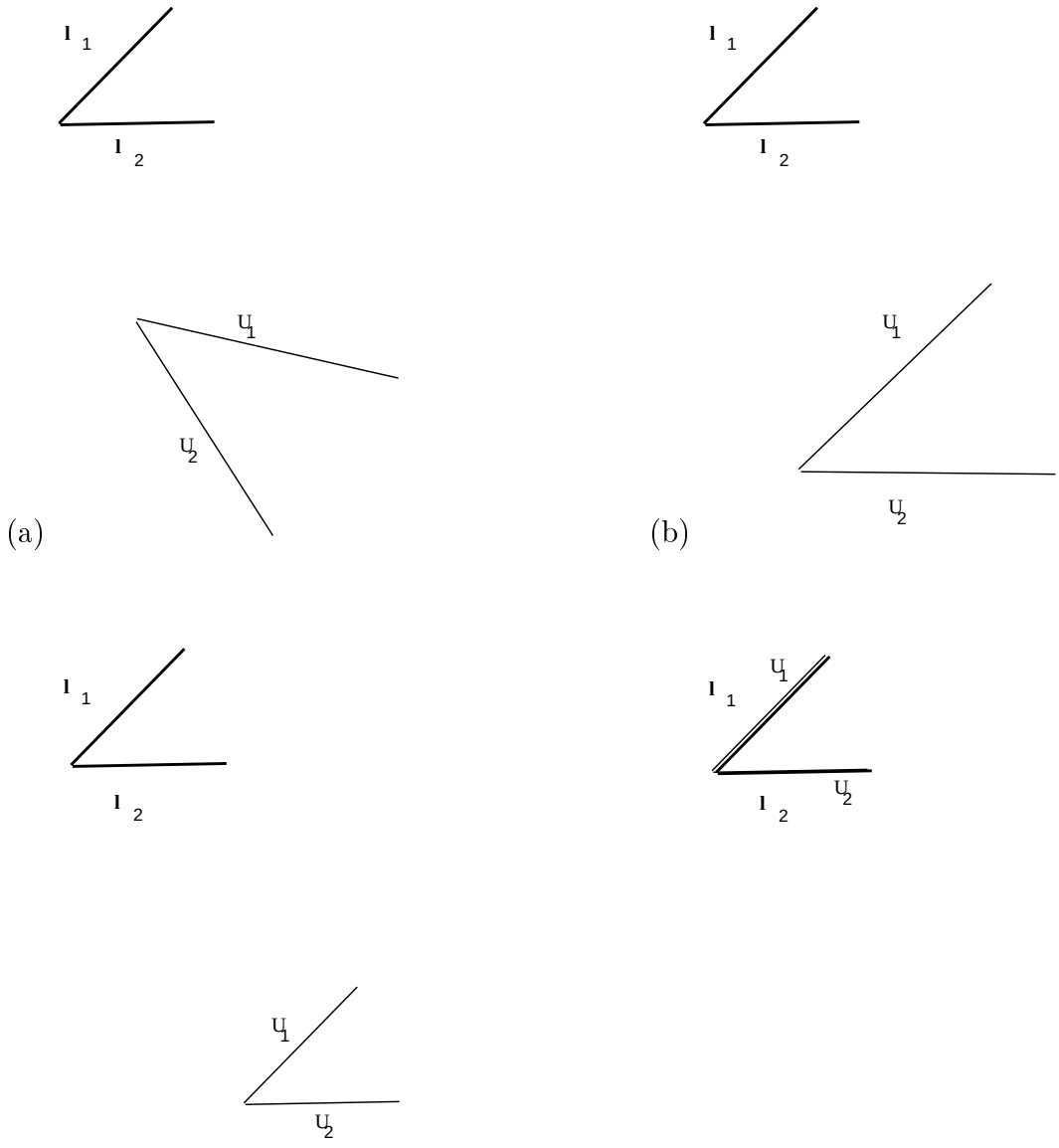


Figure 4.10 Stages in pose estimation from lines l_1 and l_2 in the image which match edges $\mathbf{u}_1$ and $\mathbf{u}_2$ respectively in the model: (a) the viewing direction $\mathbf{v}$ estimated as the intersection between the curve for the ratio and the angle; (b) the angle δ is computed; (c) the scale s is computed; (d) the translation $\mathbf{t}$ is computed.

regions, and the regions associated with compatible hypotheses should have a non-empty intersection.

We measure the position of three vertices in the ratio/angle case and the position of four vertices in the angle/angle case. Assuming that each of the measured coordinates has a normal distribution with the measured value as its mean and with a certain variance σ^2 (which can be found empirically), this induces a probability density function on the pose of the object due to these features. Thus, a multivariate normal distribution of the values measured in the image induces a distribution on the pose space.

When trying to determine whether two feature sets are part of one instance of a certain model we use the poses recovered from them. In order to account for uncertainty we have to calculate the probability that the two poses are close enough. This will be discussed in Sect. 4.7. Here we describe how to eliminate most pairs of poses whose matching probability is negligible by estimating the pose uncertainty regions and checking if they intersect.

We bound the vertex position uncertainty by a circle of radius $k\sigma$. The larger the circle the higher the probability that the real value is within that circle. Under that assumption we compute the size of the region of the pose space which could generate the feature set. If the regions for both hypotheses have an empty intersection the hypothesis pair is discarded. A large value of k increases the probability that two random hypotheses will have a non-empty intersection. Therefore there is a tradeoff in setting k such that most correct pairs will be found and a maximal number of random pairs will be discarded.

We divide the computation into two stages. In the first stage we compute the uncertainty in the values measured in the image such as ratios, angles and lengths due to the uncertainty in vertex position. This stage does not depend on the model features matched to the image feature set. In the second stage we combine the results of the first

stage with the uncertainty due to interrelations between the various components of the pose to produce the estimated pose uncertainty region.

In the first stage we find the maximum and minimum values for the various values measured in the image assuming the uncertainty is bounded by $\epsilon = k\sigma$. For the ratio/angle case (Figure 4.11(a)) given vertices P_0,P_1 and P_2 we find P_0',P_1' and P_2' within their respective uncertainty regions which yield extreme values for ratios angles or lengths. As the uncertainty is relatively small we can assume that the first derivative component of the Taylor expansion of the functions we are trying to maximize dominates the value of the function. Therefore we can assume that the extreme values happen on the boundary of the uncertainty region, and we only need to find three angles ϕ_i such that $P_i' = P_i + (\epsilon\cos(\phi_i), \epsilon\sin(\phi_i))$. To demonstrate the general approach we shall deal with the case of finding extreme ratios between the length of the lines l_1' and l_2'. We define $F(\phi_0, \phi_1, \phi_2) = (|l_1'|/|l_2'|)$. The maximum (minimum) is obtained when

$$\frac{\partial F}{\partial \phi_0}(\phi_0, \phi_1, \phi_2) = \frac{\partial F}{\partial \phi_1}(\phi_0, \phi_1, \phi_2) = \frac{\partial F}{\partial \phi_2}(\phi_0, \phi_1, \phi_2) = 0.$$

At first it seems that a three-parameter optimization must be performed to obtain the maximum (minimum) ratio. However when given a value for ϕ_0, the values for ϕ_1 and ϕ_2 can be computed directly. For example when trying to find the maximum ratio we have to extend $P_1 - P_0'$ and contract $P_2 - P_0'$ as much as possible. The maximum change for both lines which is ϵ is achieved by extending $P_1 - P_0'$ by ϵ and by cutting ϵ off $P_2 - P_0'$. For a minimum ratio $P_2 - P_0'$ is extended and $P_1 - P_0'$ is contracted. In both cases ϕ_i is chosen such that $P_i - P_0'$ is parallel to $P_i' - P_i$. Thus we are left with a one-dimensional optimization which can be easily done using standard numerical techniques such as the secant method or a combination of the secant method and the Newton-Raphson method [48]. In Table 4.1 we summarize how to find the extreme values for all the values which we are looking for. Examples of the corresponding configurations are shown in Figure

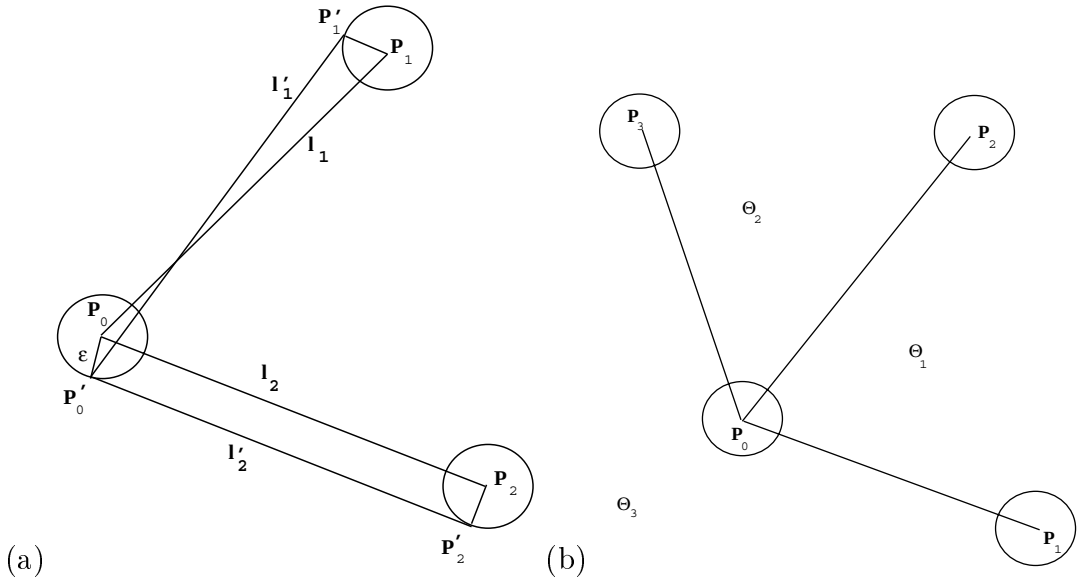


Figure 4.11 Given vertices P_i in the image, find P_i' within their respective uncertainty regions which yield extreme values for ratios, angles or lengths: (a) the two-edge case; (b) the trihedral corner case.

4.12. For the case of the trihedral corner (Figure 4.11(b)) we measure the minimum and maximum values for $\theta_1, \theta_2, \theta_3 = 2\pi - (\theta_1 + \theta_2)$ and the rotation.

In the second stage of the computation, components of the pose region are computed. When computing the projection of a model point, transformations due to $\mathbf{v}, \alpha, s$ and $\mathbf{t}$ are applied to the point in that order. Therefore, uncertainty in the initial components of the projection can increase the uncertainty of the other components of the pose. The

Value	Function	Computing ϕ_1 and ϕ_2
Ratio	$\|\|l_1'\|\|/\|\|l_2'\|\|$	$(P_i - P_0') \parallel (P_i' - P_i)$
Angle	$cos^{-1}((l_1' \cdot l_2')/(\|\|l_1'\|\|\|\|l_2'\|\|))$	$(P_i' - P_i) \perp (P_i' - P_0')$
Scale	$(\|\|l_1'\|\|/\|\|l_1\|\| + \|\|l_2'\|\|/\|\|l_2\|\|)/2.0$	$(P_i - P_0') \parallel (P_i' - P_i)$
Rotation	$Bis^{\perp} \cdot Bis'$ where Bis is the normalized bisector	$(P_i' - P_i) \perp (P_i' - P_0')$
Translation	Maximal value is ϵ	$\phi_i = \phi_0$

Table 4.1 A table describing the different values which we would like to maximize (minimize), the function which is maximized (minimized) and how to compute ϕ_1 and ϕ_2 given ϕ_0.

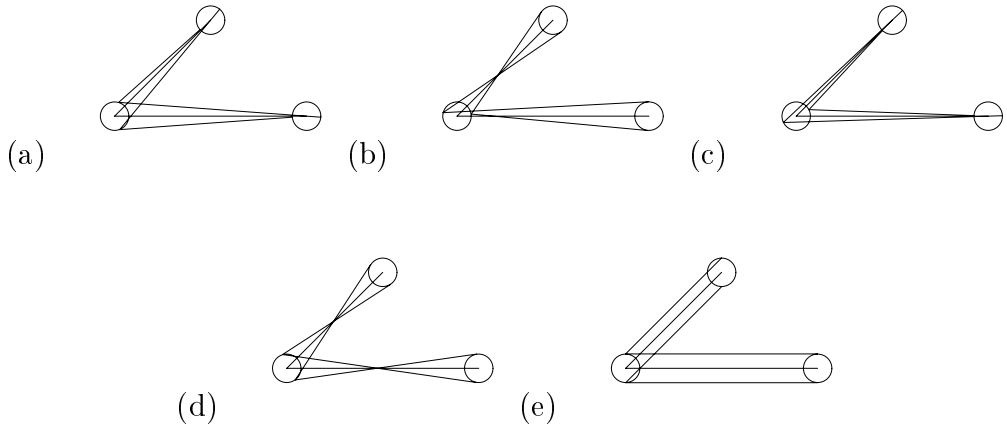
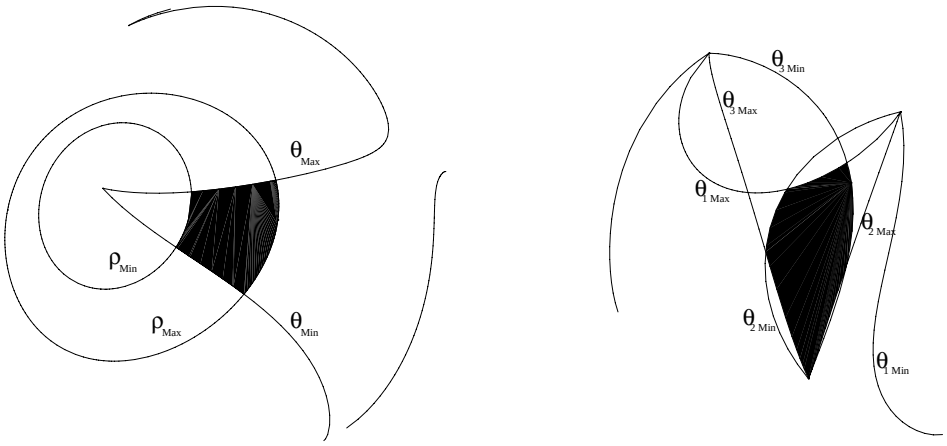


Figure 4.12 The cases where minimal and maximal values are measured. Bounded uncertainty in vertex position is assumed. (a) ratio between the lengths of edges; (b) angle between the edges; (c) length of edges; (d) rotation of the bisector of the two edges; (e) translation of the feature set in the y direction.

uncertainty of α and s is affected by the uncertainty in $\mathbf{v}$, and the uncertainty in $\mathbf{t}$ is affected by the uncertainty in $\mathbf{v}, \alpha$ and s.

For the two-edge case the region of the viewing sphere which belongs to the region is the region bounded by the curves for $\rho_{min}, \rho_{max}, \theta_{min}$ and θ_{max} which were computed in the previous step (Figure 4.13(a)). Using the boundary of the region we find the minimal and maximal values for the position of one of the vertices, the rotation, and the scale of the projected features. Using the extreme values for the the rotation α and scale s from both stages, we compute a worst case estimate for them. In order to compute the translation component we have to take into account the effects of all previous parts of the pose. We start with the region denoted by the extreme polar coordinates of the projection of a model feature point due to the viewing direction. Rotation affects the angle part of the region and scale affects the length part of the region. To that we add ϵ the translation uncertainty in the image and get the translation part of the region.



(a) (b)

Figure 4.13 The uncertainty region on the viewing sphere bounded by curves for the possible minimal and maximal values measured in the image: (a) the two-edge case; (b) the trihedral corner case.

For the case of the trihedral corner we compute the boundary of the region on the viewing sphere bounded by curves for the minimum and maximum values of θ_1, θ_2 and θ_3 (Figure 4.13 (b)). As in the previous case we estimate the uncertainty for the rotation. Due to lack of information, the components of scale and translation can not be estimated independently. Therefore for each possible value of the scale we must estimate the translation uncertainty.

To demonstrate the variability of the pose uncertainty for different model feature sets, we tabulate the pose uncertainty for them in Table 4.2 where the ratio and the angle in the image were 1.4 and 315° respectively. Nearly all the components of the pose region do not change much between the different examples. Only the size of the viewing sphere component changes dramatically for the different examples. The closer the ratio and angle measured in the model are to the values measured in the image the larger the viewing sphere component is.

Ratio	Angle	Viewing Sphere	Rotation	Scale	Translation
1.0	280.0°	0.011	0.092	0.139	0.0080
0.3	90.0°	0.001	0.101	0.147	0.0329
1.4	315.0°	0.085	0.184	0.158	0.0459
0.3	315.0°	0.001	0.109	0.173	0.0099
1.4	90.0°	0.009	0.124	0.114	0.0142

Table 4.2 Tabulation of the fractions of components of the uncertainty pose region for a given image feature set whose ratio and angle are 1.4 and 315.0° respectively for several model feature sets with different ratios and angles.

This phenomenon, which is another aspect of the probabilistic peaking effect can be exploited for solving the following problem studied in [32]. Given a number of probable correspondences between points in the image and points in a model, choose a subset of the correspondences to compute the best pose estimate. For each triple the smaller the size of the region, the smaller the uncertainty in the pose is. For larger point subsets the intersection of pose regions can be computed and the subset which yields the smallest pose region is chosen.

4.6.3 Efficient Pose Uncertainty Estimation

In the previous section we described how to estimate pose uncertainty regions for match hypotheses. Although this technique gave us insight into the nature of these regions, a more efficient technique is needed to estimate these regions at the time of recognition and find the intersection between the regions.

We compute the pose p of the object assuming a match hypothesis h is computed using the technique described in Section 4.6.1. p is a function of the vertex positions $\mathbf{a}$ measured in the image. Using Taylor expansion, the effect of a small uncertainty ξ in $\mathbf{a}$ on the pose can be estimated by:

$$p(\mathbf{a} + \xi) \approx p(\mathbf{a}) + \nabla p(\mathbf{a})\xi.$$

As the uncertainty is small the contributions of higher derivatives of the pose function can be neglected. For the ratio/angle case all six components of the pose are recovered using three coordinate pairs. Thus $\nabla p(\mathbf{a})$ is a 6×6 matrix. Assuming that the uncertainty for each coordinate pair is bounded by ϵ, the maximum uncertainty for a component p_ν of pose will be when ξ has the following values:

$$\begin{cases} \xi_{2i-1} = \epsilon \frac{\partial p_\nu}{\partial a_{2i-1}} \left(\frac{\partial p_\nu}{\partial a_{2i-1}}^2 + \frac{\partial p_\nu}{\partial a_{2i}}^2 \right)^{-1/2}, \\ \xi_{2i} \quad = \epsilon \frac{\partial p_\nu}{\partial a_{2i}} \left(\frac{\partial p_\nu}{\partial a_{2i-1}}^2 + \frac{\partial p_\nu}{\partial a_{2i}}^2 \right)^{-1/2}, \end{cases}$$

where i denotes the i^{th} coordinate pair. Thus for each coordinate pair the vector (ξ_{2i-1}, ξ_{2i}) points in the direction of $\nabla p_\nu(\mathbf{a}_{2i-1}, \mathbf{a}_{2i})$. The derivatives are computed numerically. The perturbed pose is computed using the multivariate Newton-Raphson algorithm with the unperturbed pose given as the initial guess. For most components of the pose computing the uncertainty is simple. However for the viewing direction component $\mathbf{v}$ of the pose we parameterize the pose as $\mathbf{v} = \alpha\mathbf{v}_1 + \beta\mathbf{v}_2 + \gamma\mathbf{v}_3$ where $\mathbf{v}_1$ is the viewing direction for the unperturbed input, $\alpha = \sqrt{(1-\beta^2-\gamma^2)}$, and the uncertainty is measured in radians in the $\mathbf{v}_2$ and $\mathbf{v}_3$ directions. So if the viewing direction uncertainties are $\arcsin\beta$ and $\arcsin\gamma$, the viewing direction component in the pose uncertainty region is:

$$\{\mathbf{v} \in S^2 : \quad |\mathbf{v}\cdot\mathbf{v}_2| < \beta, \quad |\mathbf{v}\cdot\mathbf{v}_3| < \gamma\}.$$

In order to check if two pose uncertainty regions have a non-empty intersection, all components of the pose are compared. This works for pairs of ratio/angle hypotheses, but for angle/angle hypotheses only the viewing direction and rotation components can be recovered directly and more information is needed to recover the scale and translation components. By adding to the feature set the position and uncertainty of the corner of the other feature set, the scale and translation components of the uncertainty region are recovered.

4.7 Ranking Recognition Results

4.7.1 Requirements

In the final stage of the algorithm, pairs of hypotheses whose pose uncertainty regions have a non-empty intersection are ranked by probability. For a ranking scheme to be useful it should exhibit the following characteristics: using the notion of "maximum likelihood interpretations" discussed in [92], more likely interpretations (hypotheses with larger pose uncertainty regions) should be ranked higher than less likely ones, interpretations which would assume a larger uncertainty in vertex position should be ranked lower than interpretations with smaller uncertainty, and feature combinations with many plausible interpretations (e.g., features belonging to a single rectangular or triangular face) should be ranked lower than feature combinations with a unique interpretation. Our probabilistic expression accounts for all these sometime conflicting requirements in ranking possible interpretations. In addition, the algorithm should be able to rank the correct hypotheses first even if the algorithm has to be stopped for lack of time before all pairs of hypotheses have been tested.

4.7.2 Derivation

Given a set of image features which participate in a match hypothesis (ratio/angle or angle/angle), the pose uncertainty region bounds the region in the pose space in which the error is bounded by ϵ. The higher the value of ϵ the higher the probability that if the hypothesis is correct, that the pose of the object lies within the uncertainty region. ϵ is set large enough such that the probability that the correct pose is not within the uncertainty region is very small.

Given two feature sets in the image e_1 and e_2 and two respective hypotheses h_1 and h_2, we define H as the hypothesis that h_1 and h_2 are true and both match image feature

sets to the same instance of a certain model M. We compute $P(h_1, h_2, H, M|e_1, e_2)$, using Bayes' rule:

$$P(h_1, h_2, H, M|e_1, e_2) = \frac{P(e_1, e_2, h_1, h_2, H, M)}{P(e_1, e_2)}. \tag{4.10}$$

For e_1 and e_2 to be features of the same object, the pose of the object p must be in the intersection of the pose uncertainty regions of the two hypotheses which we denote by $U(e_1, h_1)$ and $U(e_2, h_2)$ respectively. For each possible pose we write

$$P(e_1, e_2, h_1, h_2, H, M, p) = P(M)P_d(e_1, h_1)P_d(e_2, h_2)\chi_{U(e_1,h_1)}(p)\chi_{U(e_2,h_2)}(p)f_p(p),$$

where $\chi_{U(e_1,h_1)}(p)$ and $\chi_{U(e_2,h_2)}(p)$ are the characteristic functions of $U(e_1, h_1)$ and $U(e_2, h_2)$ respectively and $f_p(p)$ is the p.d.f. of poses in the pose space. If the position of the camera with respect to the surface on which the objects are placed is known, information about stable poses of the objects can be reflected in $f_p(p)$. By marginalizing with respect to p we obtain

$$P(e_1, e_2, h_1, h_2, H, M) = P(M)P_d(e_1, h_1)P_d(e_2, h_2) \int_p \chi_{U(e_1,h_1)}(p)\chi_{U(e_2,h_2)}(p)f_p(p)dp. \tag{4.11}$$

Assuming the poses are uniformly distributed and that the volume of the pose space is normalized to 1, (4.11) simplifies to:

$$P(e_1, e_2, h_1, h_2, H, M) = P(M)P_d(e_1, h_1)P_d(e_2, h_2)|U(e_1, h_1) \cap U(e_2, h_2)|. \tag{4.12}$$

We compute $P(e_1, e_2)$ by summing over every pair of hypotheses h_i, h_j which could generate e_1 and e_2 respectively and whether e_1 and e_2 belong to the same object ($H^{(i,j)}$) or not ($\neg H^{(i,j)}$), yielding:

$$\begin{aligned} P(e_1, e_2) = &\textstyle\sum_i \sum_j P(e_1, e_2, h_i, M^{(h_i)}, h_j, M^{(h_j)}, H^{(i,j)}) + \\ &\textstyle\sum_i \sum_j P(e_1, e_2, h_i, M^{(h_i)}, h_j, M^{(h_j)}, \neg H^{(i,j)}), \end{aligned}$$

where $M^{(h_i)}$ is the model to which the model features of h_i belong. In the first term e_1 and e_2 are feature sets of the same instance of a certain model, so the probabilities are computed in the same way that as the numerator of (4.10). In the second term, e_1 and e_2 do not belong to the same object, therefore

$$P(e_1, e_2, h_i, M^{(h_i)}, h_j, M^{(h_j)}, \neg H^{(i,j)}) = P(e_1, h_i, M^{(h_i)})P(e_2, h_j, M^{(h_j)})P(\neg H^{(i,j)}).$$

When h_i and h_j represent features belonging to different models, $P(\neg H^{(i,j)}) = 1$. If not, $P(\neg H^{(i,j)})$ is determined empirically by computing the probability that more than one instance of M_i will appear in the image. We use same arguments that were used to derive (4.11) to compute $P(e_1, h_i, M^{(h_i)})$, by taking the volume of one uncertainty region instead of the intersection of two regions, yielding:

$$P(e_1, h_i, M^{(h_i)}) = P(M^{(h_i)})P_d(e_1, h_i)|U(e_1, h_i)|. \tag{4.13}$$

Combining all these results we are able to compute $P(h_1, h_2, H, M|e_1, e_2)$.

This derivation can be easily extended to more than two hypotheses. In the numerator of (4.10) we compute the intersection of the uncertainty regions of all the hypotheses and in the denominator we sum over all possible interpretations of the feature sets under consideration (all the features belong to the same object, some of them belong to one others to another etc...). We use the results for all subsets of the set of features in order to compute that expression. It is important to note that only sets of features that all of their subsets have non-zero rank might have a non-zero rank themselves. Therefore, we only consider the small number of pairs of hypotheses which have been found by the pair-ranking procedure as input for the extended procedure, which can be performed at minimal computational cost but have very statistically significant results.

During recognition, for every pair of hypotheses whose pose uncertainty regions intersect we evaluate (4.10). Terms similar to $P(e_1, e_2, h_i, h_j, M, H^{(i,j)})$ appear in the numerator and the denominator of the expression. For hypothesis pairs whose pose un-

certainty region do not intersect, this term will be zero. Therefore, when we compute the rank of a hypothesis pair we can assume at first that all the terms of that type except $P(e_1, e_2, h_1, h_2, H)$ are zero. When we compute the rank of another hypothesis pair h_1' and h_2' which interpret the same image feature sets, we will add the value we computed for $P(e_1, e_2, h_1', h_2', M', H)$ to the denominator of the rank of h_1 and h_2. Terms of the type $P(e_1, h_i, M^{(h_i)})$ which also appear in the denominator only involve one hypothesis, therefore their value can be precomputed and stored in the look-up tables. However that is not always necessary because as the following calculation will show, their values may be very small and their impact on the value of (4.10) is negligible. We determine whether to neglect these terms by analyzing the relative sizes of terms of the type $P(e_1, e_2, h_i, h_j, M, H^{(i,j)})$ and $\sum_{i,j} P(e_1, e_2, h_i, M^{(h_i)}, h_j, M^{(h_j)}, \neg H^{(i,j)})$ and when the former is much bigger than the latter, the latter can be discarded.

We estimate $|U(e_1, h_1) \cap U(e_2, h_2)|$, in order to estimate the value of $P(e_1, e_2, h_i, h_j, M, H^{(i,j)})$. Consider the case of the one dimensional pose space and that $U(e_1, h_1)$ and $U(e_2, h_2)$ are segments of length l which overlap. When the relative positions of $U(e_1, h_1)$ and $U(e_2, h_2)$ are uniformly distributed, the average length of the overlap between them will be $l/2$. Generalizing this to the six-dimensional pose space, we estimate that $|U(e_1, h_1) \cap U(e_2, h_2)| \approx |U|/2^6$, where $|U|$ is the average volume of a pose uncertainty region. Consider the recognition system with a database of m models. Each model has on average n feature sets, and on average k of them appear in a given scene. We can estimate $P(M) \approx k/m$. Thus we can estimate that

$$P(e_1, e_2, h_i, h_j, M, H^{(i,j)}) \approx 2^{-6}|U|(k/m)P_d(e, h)^2,$$

and

$$\sum_{i,j} P(e_1, e_2, h_i, M^{(h_i)}, h_j, M^{(h_j)}, \neg H^{(i,j)}) \approx (\frac{k}{m})^2 (nm)^2 |U|^2 P_d(e, h)^2 = k^2 n^2 |U|^2 P_d(e, h)^2$$

The ratio between these two values yields:

$$\frac{P(e_1,e_2,h_i,h_j,M,H^{(i,j)})}{\sum_{i,j} P(e_1,e_2,h_i,M^{(h_i)},h_j,M^{(h_j)},\neg H^{(i,j)})} \approx \frac{2^{-6}|U|(k/m)P_d(e,h)^2}{k^2n^2|U|^2P_d(e,h)^2} \approx \frac{1}{2^6kn^2m|U|}. \tag{4.14}$$

Evaluating (4.14) for a recognition system such that $k \approx 1$,$n \approx 10$ and $|U| \approx 10^{-7}$, gives $1500/m$. Only when the number of models in the database $m > 100$ will the contribution of $\sum_{i,j} P(e_1,e_2,h_i,M^{(h_i)},h_j,M^{(h_j)},\neg H^{(i,j)})$ to (4.10) be significant, for a smaller database this term can be neglected.

The recognition algorithm traverses the list of the pairs of hypotheses in decreasing probability order. We compute the rank of pairs of hypotheses whose pose uncertainty regions have a non-empty intersection by first evaluating (4.12) and dividing it by the sum of all the values of (4.12) computed for all the pairs of hypotheses found so far which suggest interpretations to the same pair of features.

4.7.3 Characteristics of the Ranking Scheme

In section 4.7.1, we made several requirements of our ranking scheme. Here we will analyze the algorithm to see how it satisfies these requirements.

We have required that "popular" features which yield many possible interpretations (e.g., features which belong to a rectangular or triangular face) be ranked lower than features which yield few interpretations. "Popular" features will participate in many hypothesis sets. Therefore their corresponding value of $P(e_1,e_2,h_1,h_2,H,M)$ will contribute to the denominators of the probabilities of all the interpretations, thus reducing the ranks of them all. This is reasonable because this set of features does not allow us to discriminate between the different hypotheses, where-as a less "popular" but probably correct set of features will have a higher rank, since not many competing hypotheses will exist for that set of features.

We have required that even if the algorithm had to be stopped without checking all pairs of hypotheses, the correct recognition results would be ranked high on the list. As the hypotheses are traversed in decreasing probability order there is a high probability that the correct hypothesis pairs will be ordered high on the list. Therefore, if the recognition process has to be interrupted, we can still assume that the match hypotheses corresponding to the correct interpretation have been processed. We are especially interested in the the "non-popular" feature sets. For them the correct interpretation has been found and there is a small chance that any other competing interpretations would have been found even if all pairs of hypotheses had been checked. Therefore most "non-popular" hypotheses will have a high rank and that rank would be equal to the final rank in many cases.

We also made several requirements on how competing interpretations for the same set of features should be ranked. We demonstrate the performance of the ranking algorithm using illustrations of pose uncertainty regions of typical hypothesis pairs showed in Figure 4.14.

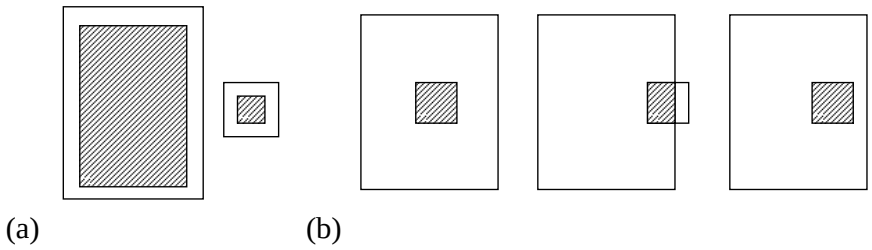


Figure 4.14 Illustration of the performance of the ranking scheme: (a) the algorithm ranks higher a pair of likely hypotheses with large pose uncertainty regions (left) than a pair of unlikely hypotheses with small pose uncertainty regions (right); (b) the algorithm ranks higher a hypothesis pair with a small error in vertex position (left) over a pair with a large error (middle) but can not prefer the pair on the right to the pair on the left even though the error for that pair is larger.

The algorithm ranks competing interpretations for a pair of features by comparing the intersection of the pose uncertainty regions of the two pairs of hypotheses. In Figure

4.14(a), a pair of likely hypotheses with large pose uncertainty regions (left) are ranked higher than a pair of unlikely hypotheses with small pose uncertainty regions(right). Thus the algorithm would prefer the "maximum likelihood" interpretation [92] over the less likely interpretation. In Figure 4.14(b), we study the case in which the size of the pose uncertainty regions is the same but their relative positions are different. The closer the centers of the regions are, the smaller the error in vertex position will be if the interpretation is correct. The algorithm ranks higher the hypotheses pair with a small error in vertex position(left) over a pair with a large error(middle) which causes their uncertainty regions to not fully intersect. The algorithm however does not rank the pair on the left higher than the pair on the right even though the pair on the right assumes a larger uncertainty in vertex position because the size of the intersection of the pose uncertainty regions is the same. In the next section we present a variant of the ranking scheme which addresses this problem.

4.7.4 Exact Ranking Scheme

The fundamental characteristic of the algorithm which prevents it from discriminating between the two interpretations illustrated in Figure 4.14(b)(left,right) is that all poses within the intersection of the pose uncertainty regions have equal weight even though the poses which yield small vertex position errors should have a higher weight than poses which yield large errors.

To solve this problem we will weight each pose by the distance between the image features and the model features backprojected using that pose. Assuming the uncertainty in vertex position has a normal (or any other known) distribution, we use the probability density function value for the computed vertex position uncertainty as the weight for

that pose. Substituting this expression into (4.11) yields:

$$P(e_1, e_2, h_1, h_2, H, M) = P(M)P_d(e_1, h_1)P_d(e_2, h_2) \int_p f_1(p) f_2(p) f_p(p) dp, \tag{4.15}$$

where $f_1(p)$ and $f_2(p)$ denote the probability density functions applied to the errors in e_1 and e_2 respectively assuming the pose is p.

Similarly (4.13) is transformed into:

$$P(e_1, h_i, M^{(h_i)}) = P(M^{(h_i)})P_d(e_1, h_i) \int_p f_1(p) f_p(p) dp.$$

This ranking scheme correctly ranks 4.14(b)(left) higher than 4.14(b)(right). In order to use this scheme we would have to evaluate expressions of the type (4.15) during recognition time. There is no closed form solution for evaluating integrals of that type and numerical Monte-Carlo integration techniques must be used. These techniques are computationally very costly so we recommend using the simpler recognition ranking scheme presented in Section 4.7.2.

4.8 Experimental Recognition Results

In this section we present the implementation of our recognition algorithm and show experimental results of running it on real images. Our model database consists of the five objects shown in Figure 4.15.

We extract edges from the image using the Canny edge detector [16] and then detect lines from the extracted edges. We combine these lines automatically into feature sets using the following technique: we detect corners in the image as the intersection point of the supporting lines of two lines in the image when the actual termination points of the two lines is close to the intersection point. Once a corner has been detected, additional lines which terminate close to the corner are added to the list of lines emanating from

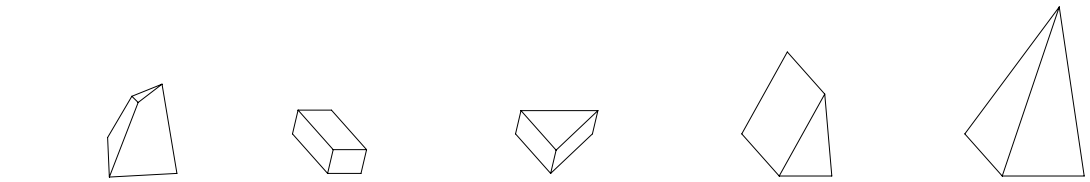


Figure 4.15 Model database: (a) a truncated pyramid, (b) a box, (c) a triangular prism, (d) another prism, (e) a pyramid.

it. We label lines which end in the middle of another line (T junctions) as partially occluded edges. We generate feature sets from this information. For each line triple emanating from a corner, we generate an angle/angle feature set. For each pair of lines emanating from a corner, we generate an ratio/angle feature set when both lines start and end at a vertex, or an occluded ratio/angle feature set when one or both of them end at a T junction. Since in the current implementation none of the models in our database displays self occlusion, occluded ratio/angle feature sets are discarded. The feature sets extracted in this stage need not be perfect because an important feature of our recognition algorithm is that it is robust to uncertain and incomplete input.

For each feature set, we retrieve from the look-up tables the corresponding match hypotheses and their probabilities. The match hypotheses for all the feature sets are combined into a list, sorted by probability and processed in that order. For each hypothesis, we search for matches with compatible pose uncertainty regions. We compute the rank of each compatible pair of matches found and maintain a list of compatible pairs of matches sorted by rank. The algorithm outputs the interpretations due to the pairs of matches with the highest ranks. As was explained in Section 4.7.2, this raking scheme can be extended to deal will larger sets of features with a small extra computational cost. In this implementation however, only pairs of matches were considered.

To make this algorithm a complete recognition system the following two steps have to be added: least squares estimation of pose, and hypothesis verification by backprojection.

These two steps have not been implemented yet since the focus was put on finding efficient ways to generate promising hypotheses.

We now present several examples of results obtained running our algorithm on real images.

Figure 4.16(a) shows an image of the rectangular pyramid and the second prism. Note that the results of the edge detection and line extraction (Figure 4.16(b)) contain features that are due to the background and shadows. Figure 4.16(c) shows the features belonging to feature sets extracted from the image. Note that the line due to the shadow of the prism is part of angle/angle feature sets where the other lines participating in those feature sets are edges of the prism itself. Edges which have not been fully extracted by the edge detector may only participate in angle/angle feature sets and are discarded when they are not adjacent to a trihedral corner. Applying this criterion to this image caused most features due to the background to be discarded. The objects recognized by the algorithm are shown in Figure 4.16(d).

Results of running the algorithm on an image of two triangular prisms with partial occlusion are shown in Figure 4.17. Note that the edge detector was not able to recover the internal edge of the triangular shape but found all the silhouette edges.

In Figure 4.18, the results of processing another image of the two triangular prisms with partial occlusion are shown. Here, a different part of the second prism is occluded. Note that again internal edges were not detected by the edge detector. The internal edge of the second prism is especially interesting. Parts of the edge were detected but not the whole edge. Therefore the segments of the edge participate only in angle/angle feature sets but not in ratio/angle feature sets.

We collected in Table 4.3 statistics regarding the run of the algorithm on the examples shown above. For each run, we tabulated the number of feature sets extracted from the image, the number of match hypotheses retrieved from the look-up tables, the number of

pairs of match hypotheses which were tested for compatibility, the number of compatible pairs found, how many of them were correct and how many of the first ten compatible pairs rank by probability were actually correct and their ranks.

The ranking of the possible interpretations appears quite good. Only the results for the image in 4.16 seem disappointing. The reason for this is symmetry: all the objects which appear in the images are symmetric. However, only the pyramid has a four-way symmetry whereas the others have only a two-way symmetry. For a symmetric object there are a number of correct poses which yield the same image. Therefore the probability of the correct interpretation is divided between the matches due to the symmetric poses, reducing the ranks of them all. When the symmetric hypotheses were combined (by hand) into one, the results changed dramatically and nine of the first ten results were correct. We can exploit symmetry further by removing symmetric match hypotheses from the match hypothesis look-up tables, reducing considerably the number of match hypotheses and hypothesis pairs which have to be processed. We can automatically detect the symmetries of an object using the method proposed by Flynn [30].

In Table 4.4, we display the running times of the algorithm on the three images. For each run we tabulated the time it took the algorithm until a correct hypothesis pair of one of the two objects has been tested, the time until a correct hypothesis pair belonging to the other object is tested and the total run time of the algorithm which terminates after testing all hypothesis pairs. The algorithm was implemented in Lisp and run on a Sun SPARC10 computer. It is important to note that due to the rank of the match hypotheses most of the correct results will be found early in the run of the algorithm and there is no need to test all possible hypothesis pairs. Also note that our main concern was to explore the concepts underlying these algorithms, and little effort was made to implement this algorithm efficiently.

Figure	Feature Sets	Match Hypotheses	Hypothesis Pairs	Possible Results	Correct Results	Correct Results of the 10 Highest Ranked Results
4.16	25	3134	48428	184	88	3 (1,2,8)
4.17	11	1516	4849	55	11	5 (1-3,7,8)
4.18	16	1980	15672	140	42	9 (1,3-10)

Table 4.3 The recognition results table shows statistics from various stages of the recognition algorithm.

Figure	Time Until First Object Hypothesis	Time Until Second Object Hypothesis	Total Run Time
4.16	7min	24min	68min
4.17	3min	6min	12min
4.18	2min	3min	26min

Table 4.4 Timing of the run of the algorithm: time until the first hypothesis pair which correctly recognizes the first object was tested; time until the first hypothesis pair which correctly recognizes the second object was tested;

We show several incorrect hypothesis pairs found by our algorithm in Figure 4.19 in order to characterize them and suggest methods to avoid processing the pairs of matches which yielded them in the first place. In Figure 4.19(a), the interpretation of the scene is correct, only the pose of the object is wrong. However, as the internal edges of the object were not found by the edge detector there is no way to distinguish between the correct and incorrect pose, they both yield the same silhouette. A similar example is shown in Figure 4.19(b). However in this case internal edges were recovered by the edge detector and the correct match should be ranked higher when more features are added to the match.

The examples in Figures 4.19(c,d) show how any rectangular or triangular face can be matched to any other face of the same type. In order to avoid this type of erroneous pairs of matches, it is better not to process pairs of hypotheses whose model features belong to the same face at all. Although correct hypothesis pairs will also be discarded, the performance of the recognition algorithm will not be hurt because the algorithm could

not distinguish between the correct pair and the many incorrect ones and gives them all a low rank.

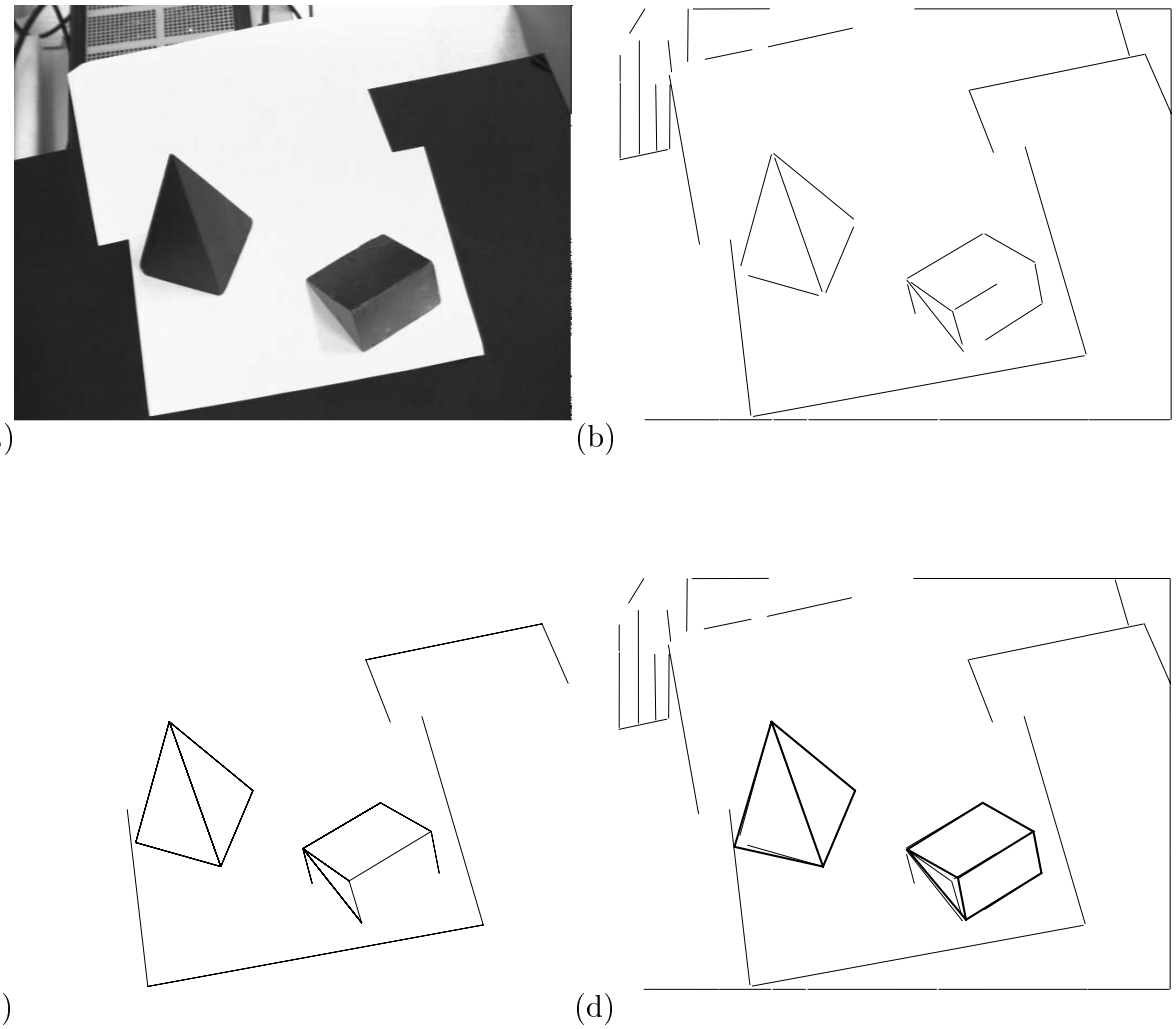
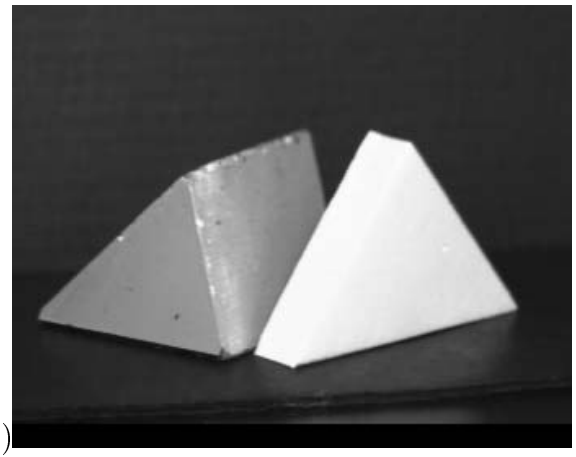
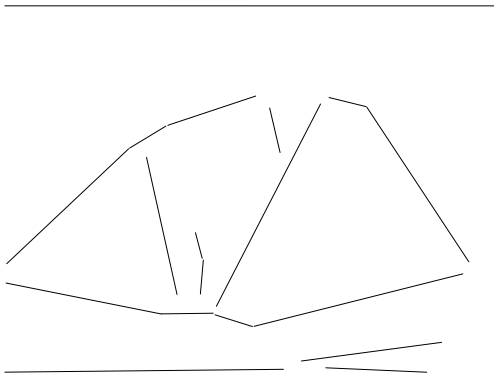


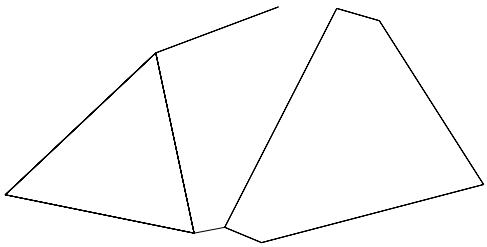
Figure 4.16 Recognition results for an image of a rectangular pyramid and a prism: (a) the image; (b) the results of edge and line detection; (c) feature sets recovered from the image; (d) recognized objects.



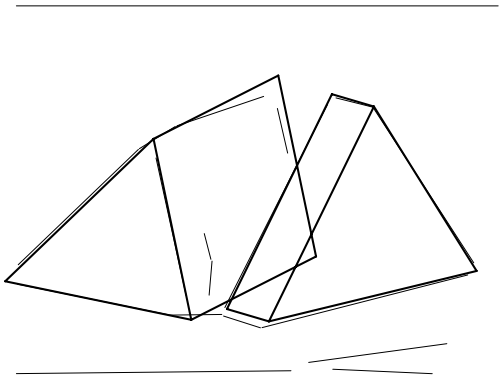
(a)



(b)



(c)



(d)

Figure 4.17 Recognition results for an image of two triangular prisms with partial occlusion: (a) the image; (b) the results of edge and line detection; (c) feature sets recovered from the image; (d) recognized objects.

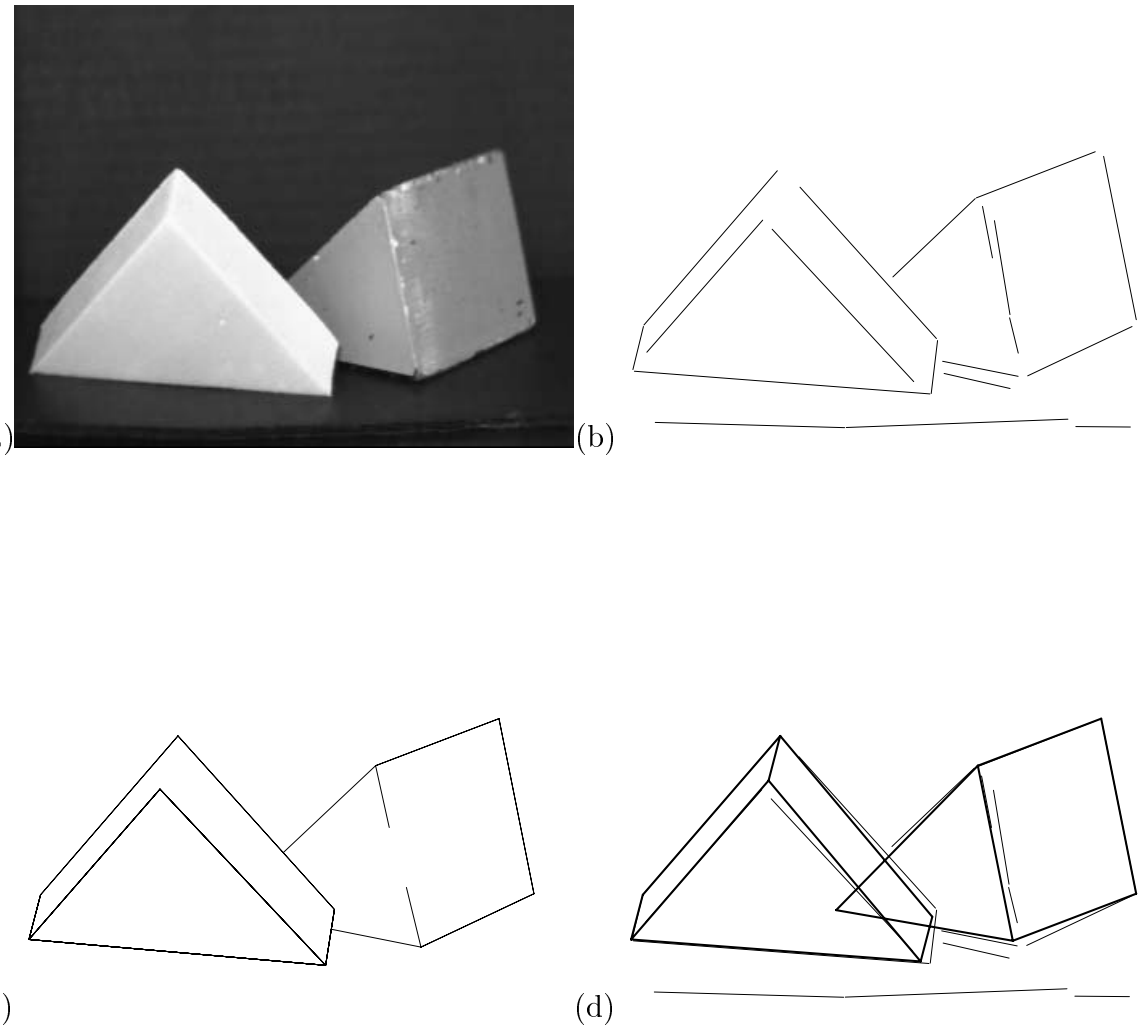


Figure 4.18 Recognition results for another image of two triangular prisms with partial occlusion: (a) the image; (b) the results of edge and line detection; (c) feature sets recovered from the image; (d) recognized objects.

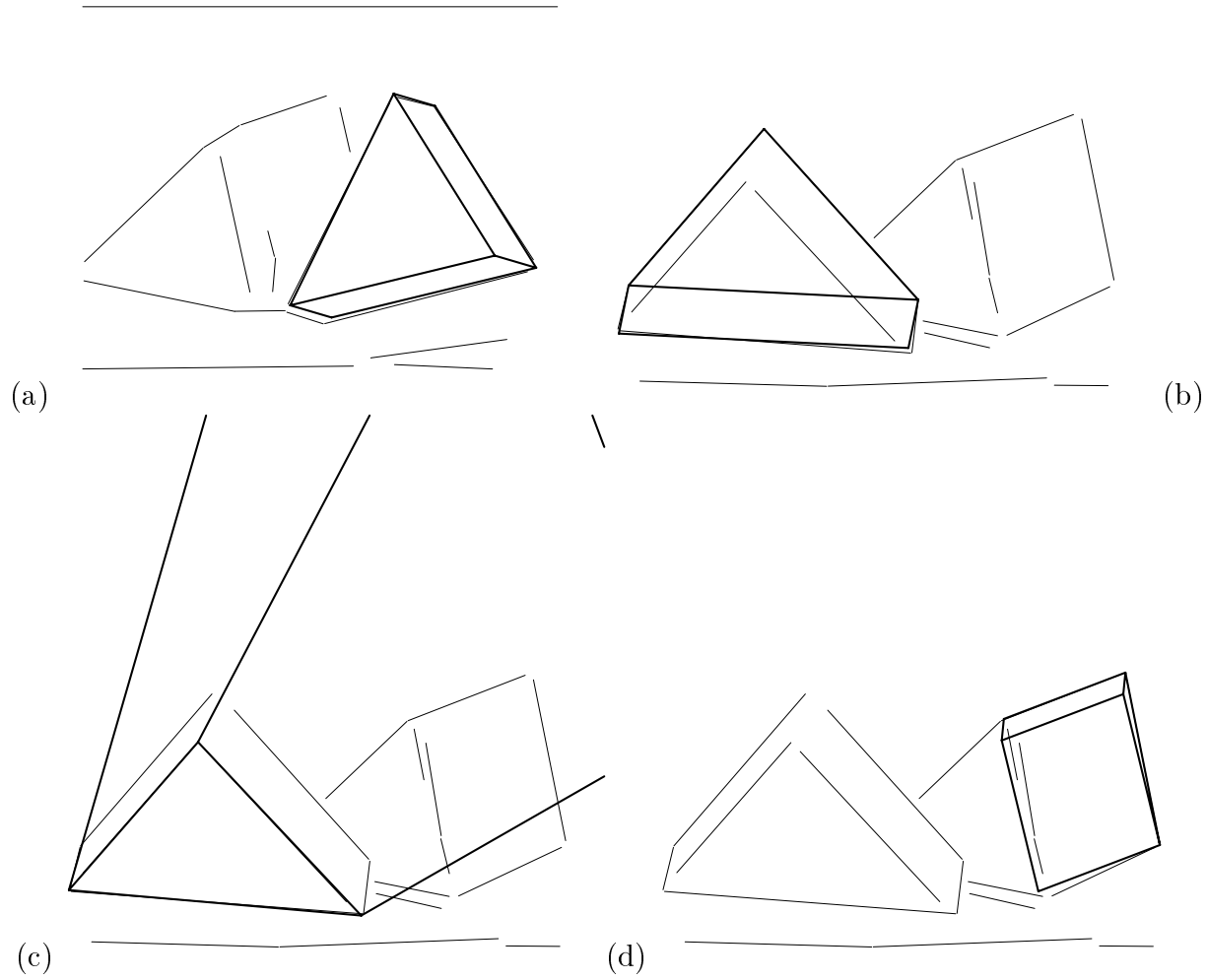


Figure 4.19 Wrong recognition results: (a) the triangular object with an incorrect pose; (b) the triangular object with another incorrect pose which contradicts some of the other features extracted from the image; (c) a triangular face of the triangular object is matched to a face of the truncated pyramid; (d) a rectangular face of the prism is matched to the wrong rectangular face of the prism.

4.9 Discussion

In this chapter we have presented a probabilistic 3D object recognition algorithm. We have studied the nature of the iso-ratio and iso-angle curves and traced them on the viewing sphere. We have used these curves to accurately compute conditional probabilities that image features match model features. The probabilities have been incorporated into a probabilistic model which takes into account the uncertainties inherent in the input to the recognition algorithm. These probabilities have been used to decide the order in which to process the match hypotheses. We have developed a method to compute the pose of the object using a minimal feature set in the image which matches to feature set in the model database. Taking the uncertainty in values measured in the image into account, we have computed the uncertainty pose region for each hypothesis. By finding hypotheses whose pose regions have a non-empty intersection, we have been able to find a set of feature sets which reinforce the recognition hypothesis. We have ranked these hypotheses by computing the probability that all the feature sets in the set came from the same instance of the suggested object. To make this algorithm a complete recognition system least squares estimation of pose, and hypothesis verification by backprojection have to be added.

Future work will be dedicated to improving the efficiency of the algorithm by finding better ways to order the rank hypotheses, process hypothesis pairs and find smaller and more accurate pose uncertainty regions. Another important research direction would be to extend this probabilistic recognition scheme to deal with more complicated objects such as curved objects.

CHAPTER 5

Summary

In this thesis we have studied various effects of uncertainty on the interpretation of images of polyhedral objects.

In Chapter 2, we have presented an algorithm which recovers the shape of 3D polyhedra using line-drawing analysis and shading information. The main contributions of this approach are that:

- it deals explicitly with uncertainty in vertex position;
- it affords a general scheme for 3D shape recovery using complex reflectance models.

The main limitation of this approach is that it assumes that the line-drawing has been correctly extracted by the edge detector. Future research should be directed to combining line-drawing analysis and shading information to correctly extract the line-drawing, fully automating the shape recovery process.

In chapter 3, we have computed the finite-resolution aspect graph of polyhedral objects. The main contributions of our approach are that:

- it provides a more realistic aspect graph which accounts for the finite resolution of the camera;

- it provides a systematic method for computing the finite-resolution view of an object;

- we have analyzed the complexity of the algorithm and the size of the aspect graph.

The main limitation of our approach is that although the finite-resolution aspect graph has the same asymptotic complexity as the classical aspect graph, its size is still larger for some type of objects but smaller for others. Future research should be directed to achieving better understanding of the complexity of aspect graphs for various classes of objects and developing more efficient algorithms for computing them. Better understanding is needed on the efficient construction of finite-resolution views.

In Chapter 4, we have developed a 3D object recognition algorithm. The main contributions of our approach are:

- we have obtained a better understanding of the probabilistic peaking effect and developed techniques for accurately computing the conditional probabilities related to that effect;

- we have developed new techniques for computing the pose of an object and estimating the pose uncertainty region;

- we have developed a probabilistic expression for ranking possible scene interpretations;

- we have integrated them into a fully-implemented recognition algorithm.

The main limitations of this approach are that it applies only to polyhedral objects, and that it would not be very efficient in very cluttered scenes and when using a large database of objects. Future research should be directed to incorporating additional information (e.g., clustering of image features or shading information) into the probabilistic

model, which would help guide the recognition process, and extending this general framework to deal with other classes of objects such as curved objects.

Although three different problems were attacked in this thesis, similar techniques were used and common problems were addressed in the different chapters.

Uncertainty in vertex position was addressed in chapters 2 and 4. In Chapter 2 uncertainty in vertex position was accounted for in checking the legality of line-drawings, and in Chapter 4 it was accounted for in estimating pose uncertainty regions and in ranking sets of hypotheses.

We have used similar techniques for tracing curves in chapters 3 and 4. In Chapter 3, they were used to trace curves of the finite-resolution aspect graph and in Chapter 4 they were used to trace iso-ratio and iso-angle curves.

In Chapter 4, aspect graphs have been computed using techniques similar to the ones used in Chapter 3. These aspect graphs have been used to compute the probability that sets of features are visible and to check in which non-critical regions of the aspect graph features were partially or fully occluded and use this information in computing the probability functions for the match hypotheses.

This thesis has attacked three problems in which uncertainty plays a key role. A better understanding of the effects of uncertainty in computer vision is needed to design more robust algorithms and more realistic representations.

APPENDIX A

Construction of Finite-Resolution Aspects

The input to the aspect simplification algorithm is an infinite-resolution aspect. The algorithm merges features which are closer than a preset distance ϵ and produces a finite-resolution aspect. The finite-resolution aspect represents the incidence relationships between the features (i.e., which vertices seem to lie on which edges), not the exact geometric position of each feature which changes for different aspects in the same critical region.

In the first stage of the algorithm the projection of the infinite-resolution aspect of the object is created using the given viewing direction. From this information and the structure of the object we create the following data structures:

- the graph which is the incidence relationships between the visible vertices and edges in the image,
- a list of pairs of close features.

The goal of the algorithm is to update the graph by recording the fact that pairs of close features lie on each other. There are various operations which can be done to incorporate pairs of features into the graph. They can be as simple as merging two vertices or putting a vertex on an edge, or more complicated as creating a "thick vertex". For each such operation there are preconditions which must be fulfilled, changes to the graph, and

the corresponding pairs of features which are removed from the list. Complex operations were defined only when their effect could not be achieved by applying a series of simpler operations. Operations are sorted by complexity, and the most complex operation whose preconditions have been met is performed. Certain operations will update the graph and remove pairs of features from the list where others will just remove features from the list which were discovered to be meaningless. This might enable a different operation that was previously disabled due to pair no removed to be executed. The process is repeated until there are no more operations that can be performed.

We now present a list of the operations, the circumstances in which they are activated, and actions they perform. Each action is accompanied by an illustration. In these illustrations we denote two close features by a dashed line and two features which are more than ϵ apart are denoted by $\leftrightarrow$.

- Two close vertices are merged in the graph (Figure A.1(a)) when no other feature is close to one of them but not to the other. The feature pair is removed from the list.

- A vertex close to an edge is placed on the edge in the graph (Figure A.1(b)) when no other feature is close to one of them but not to the other. The feature pair is removed from the list.

- A vertex which is not close to another vertex but is close to two edges incident to it, is merged with the vertex in the graph but the merged vertex is labelled as a "thick" vertex (Figure A.1(c)). The feature pairs involving the vertex and the two edges are removed from the list.

- Usually when the length of a projection of an edge is less than ϵ, its vertices are merged. When however one of the vertices is also close to another edge (Figure

A.1(d)), the edge seems to extend till that edge making it seem to be longer. In that case we place the vertex on the other edge.

- Pairs of vertices which lie on the same edge are removed from the list (Figure A.1(e)). This is the first example of an operation which does not update the graph only removes spurious pairs of features from the list such that other operations might be enabled.

- A similar but more complicated operation occurs when the pairs of vertices lie on two edges but there are two pending edge vertex pairs which would cause the two edges to be merged (Figure A.1(f)). Again spurious pairs are removed from the list and the graph is not updated.

- When several vertices are near each other they can not be merged when there exists another feature which is close to some of them but not to them all. This operation illustrated in Figure A.1(g), deals with the case where the obstructing feature is a vertex which lies on the same edge as one of the other vertices and therefore the vertices can be merged as they will all lie on the same edge at the end. The operation merges the close features and removes the appropriate feature pairs from the list.

- When the viewing direction is just above the plane of a face the width of the face which is defined as the maximum distance between a vertex and an opposite edge is less than ϵ the face is collapsed into a series of adjacent edges between the two vertices furthest apart from each other (Figure A.1(h)). All the other vertices must be near one of the edges which remain.

- A similar case occurs when all the vertices in the face a close to non-adjacent edges but not to the series of edges into which the face was supposed to collapse (Figure

A.1(i)). In this case the region occupied by the face is designated as a dark region as the whole region is populated by close edges.

- In the last case there are three edges from different faces such that the first is close to the second which is close to the third but the first and the third are not close to each other (Figure A.1(j)). The result is again a dark area.

We try to apply these operations in the reverse order to the order in which they were presented here. After an operation has been applied successfully, we attempt to apply these operations again. This is repeated until the list of pairs of close features is empty.

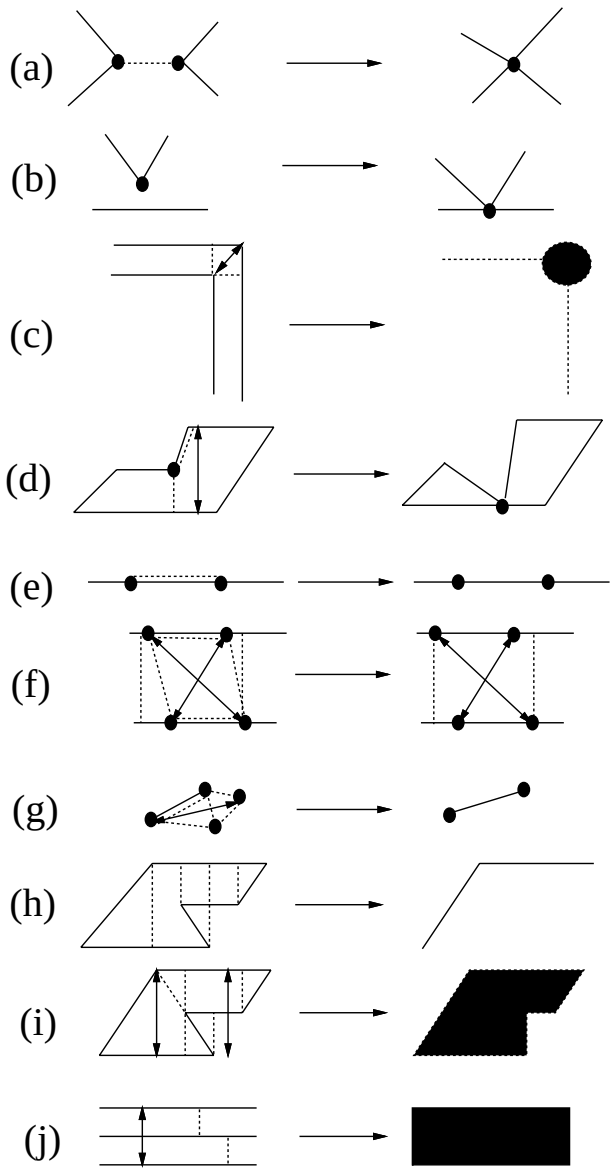


Figure A.1 Aspect simplification operations.

APPENDIX B

Curve Tracing and Homotopy Continuation

"Solving" a set of polynomial equations may entail deciding whether these equations admits common zeros, computing these zeros (they may be isolated or lie on curves or surfaces), or characterizing the regions where a set of inequalities is satisfied (cell decomposition) [5]. In this appendix, we present for completeness the set of tools that we have used for solving the first two problems. The following is adapted from [55] with permission of the authors. Symbolic and numerical methods for cell decomposition are discussed in [5, 6, 29, 55] for example.

B.1 Curve Tracing

Consider the problem of tracing a curve Γ defined implicitly in $\mathbb{R}^{n+1}$ by n polynomial equations in $n+1$ unknowns:

$$\begin{cases} P_1(X_0, X_1, \ldots, X_n) = 0, \\ \ldots \\ P_n(X_0, X_1, \ldots, X_n) = 0. \end{cases} \tag{B.1}$$

We present an algorithm that overcomes the main difficulties of curve tracing, namely finding a sample point on every real branch and marching through singularities.

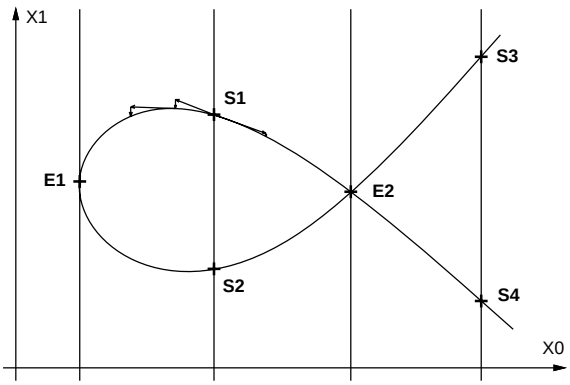


Figure B.1 Curve tracing.

The algorithm is decomposed into the following steps: (1) Compute all extremal points of Γ in some direction, say X_0 (this includes all singular points). (2) Compute all intersections of Γ with the hyperplanes orthogonal to the X_0 axis at the extremal points. (3) For each interval of the X_0 axis delimited by these hyperplanes, intersect Γ and the hyperplane passing through the mid-point of the interval to obtain one sample for each real branch. (4) March numerically from the sample points found in step (3) to the intersection points found in step (2) by predicting new points through Taylor expansion and correcting them through Newton iterations.

Figure B.1 illustrates this algorithm with a curve traced in $\mathbb{R}^2$. This curve has two extremal points $\mathbf{E}_1, \mathbf{E}_2$ and four regular branches with sample points $\mathbf{S}_1$ to $\mathbf{S}_4$; note that $\mathbf{E}_2$ is singular.

Step (1) of the algorithm requires the computation of the extrema of Γ in the X_0 direction. These points are the solutions of the system of $n+1$ polynomial equations in $n+1$ unknowns obtained by adding the equation $|J| = 0$ to system (B.1). Here, J is the Jacobian matrix $(\partial P_i / \partial X_j)$, with $i, j = 1, .., n$. Steps (2) and (3) require computing the intersections of a curve with a hyperplane, and these points are once again the solutions of a square system of polynomial equations. We use the homotopy continuation method, as described in next section, to solve these systems.

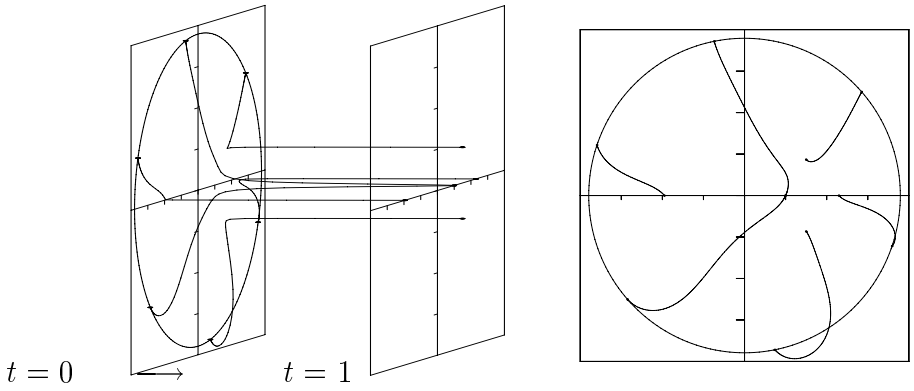


Figure B.2 An example of continuation. The right view is seen from the t axis.

The actual tracing is performed in step (4) of the algorithm. It uses a classical prediction/correction approach based on a first order Taylor expansion of the P_i's (higher order expansions could also be used [8, 27]). This involves inverting the Jacobian matrix J which is guaranteed to be nonsingular on extrema-free intervals.

Note that curve branches are connected to each other in step (4), yielding a graph structure similar to the s-graph representation of plane curves constructed through cylindrical algebraic decomposition [4]. Finally, note that all real branches can be traced in parallel. As shown below, finding the extrema of the curve and its intersections with a family of hyperplanes is a parallel process too.

B.2 Homotopy Continuation

Consider the following system of n polynomial equations in n unknowns:

$$\begin{cases} P_1(X_1, \ldots, X_n) = 0, \\ \ldots \\ P_n(X_1, \ldots, X_n) = 0. \end{cases} \tag{B.2}$$

Let us denote this system by $\mathbf{P}(\mathbf{X}) = \mathbf{0}$, with $\mathbf{P} = (P_1, \ldots, P_n)^T$ and $\mathbf{X} = (X_1, \ldots, X_n)^T$. To solve it, we use the homotopy continuation method [61], itself a simple form of curve

tracing. The principle of the method is as follows. Let $\mathbf{Q}(\mathbf{X}) = \mathbf{0}$ be another system of polynomial equations with the same total degree as $\mathbf{P}(\mathbf{X}) = \mathbf{0}$, but known solutions. A homotopy, parameterized by $t \in [0, 1]$, can be defined between the two systems by:

$$(1 - t)\mathbf{Q}(\mathbf{X}) + t\mathbf{P}(\mathbf{X}) = \mathbf{0}. \tag{B.3}$$

The solutions of the target system are found by tracing the curve defined in $\mathbb{R}^{n+1}$ by these equations from $t = 0$ to $t = 1$ according to step (4) of our curve tracing algorithm. In this case however, the sample points are the known solutions of $\mathbf{Q}(\mathbf{X}) = \mathbf{0}$ at $t = 0$, which allows us to bypass step (3) of the algorithm. It can also be shown [61] that with an appropriate choice of $\mathbf{Q}$, the curve has no extrema or singularities, which allows us to also bypass steps (1,2).

Figure B.2 shows an example of continuation, where a single equation in one unknown $P(X) = X^6 - 5.35X^5 + 7.265X^4 + 9.925X^3 - 38.7X^2 + 39.31X - 13.455 = 0$ is solved by tracking six paths in the complex plane, starting at the roots of $Q(X) = X^6 - 1$, i.e., the roots of unity, at $t = 0$, and ending at the three real roots and the two conjugate complex roots of $P(X)$ at $t = 1$. Note that one of the real roots is a double root, with two paths converging toward it.

Like regular curve tracing, continuation is a parallel process. It is actually where most time can be saved by exploiting parallelism, since all complex branches of the continuation curve must be traced. In contrast, in most applications of curve tracing, we are only interested in the (relatively few) real branches of the curves to be traced. We use a distributed implementation by Kriegman and Ponce, which runs on networks of UNIX workstations and can be used to solve systems with up to a few thousand roots.

References

[1] T. D. Alter. 3D pose from 3 points using weak-perspective. *IEEE Trans. Patt. Anal. Mach. Intell.*, 16(8):802–808, August 1994.

[2] T. D. Alter and D. W. Jacobs. Error propagation in full 3D-from-2D object recognition. In *Proc. IEEE Conf. Comp. Vision Patt. Recog.*, pages 892–898, Seattle, Washington, June 1994.

[3] R. D. Arnold and T. O. Binford. Geometric constraints in stereo vision. In *Proc. SPIE meeting*, San Diego, California, July 1980.

[4] D.S. Arnon. Topologically reliable display of algebraic curves. *Computer Graphics*, 17(3):219–227, July 1983.

[5] D.S. Arnon. Geometric reasoning with logic and algebra. In D. Kapur and J. Mundy, editors, *Geometric Reasoning*, pages 37–60. MIT Press, 1989.

[6] D.S. Arnon, G. Collins, and S. McCallum. Cylindrical algebraic decomposition I and II. *SIAM J. Comput.*, 13(4):865–889, November 1984.

[7] N. Ayache and O. D. Faugeras. HYPER: a new approach for the recognition and positioning of 2D objects. *IEEE Trans. Patt. Anal. Mach. Intell.*, 8(1):44–54, January 1986.

[8] C.L. Bajaj, C.M. Hoffmann, R.E. Lynch, and J.E.H. Hopcroft. Tracing surface intersections. *Computer Aided Geometric Design*, 5:285–307, 1988.

[9] D.H. Ballard and C.M. Brown. *Computer Vision*. Prentice-Hall, Englewood Cliffs, NJ, 1982.

[10] P. Beckmann and A. Spizzichino. *The scattering of electromagnetic waves from rough surfaces*. Pergamon, New York, 1963.

[11] J. Ben-Arie. The probabilistic peaking effect of viewed angles and distances with application to 3-D object recognition. *IEEE Trans. Patt. Anal. Mach. Intell.*, 12(8):760–774, August 1990.

[12] P.J. Besl and R.C. Jain. Three-dimensional object recognition. *ACM Computing Surveys*, 17(1):75–145, march 1985.

[13] T.O. Binford, T. Levitt, and W. Mann. Bayesian inference in model-based machine vision. In *Workshop on Uncertainty in Artificial Intelligence*, 1987.

[14] K.W. Bowyer and C.R. Dyer. Aspect graphs: an introduction and survey of recent results. *International Journal of Imaging Systems and Technology*, 2:315–328, 1990.

[15] J. B. Burns, R. S. Weiss, and E. M. Riseman. View variation of point-set and line-segment features. *IEEE Trans. Patt. Anal. Mach. Intell.*, 15(1):51–68, January 1993.

[16] J.F. Canny. A computational approach to edge detection. *IEEE Trans. Patt. Anal. Mach. Intell.*, 8(6):679–698, November 1986.

[17] G. Castore. Solid modeling, aspect graphs, and robot vision. In Pickett and Boyse, editors, *Solid modeling by computer*, pages 277–292. Plenum Press, NY, 1984.

[18] S. Chen and H. Freeman. On the characteristic views of quadric-surfaced solids. In *IEEE Workshop on Directions in Automated CAD-Based Vision*, pages 34–43, June 1991.

[19] R.T Chin and C.R. Dyer. Model based recognition in robot vision. *ACM Computing Surveys*, 18(1):67–108, January 1986.

[20] M.B. Clowes. On seeing things. *Artificial Intelligence*, 2(1):79–116, 1971.

[21] M. F. Cohen and D. P. Greenberg. The hemi-cube: A radiosity solution for complex environments. In *Proc. SIGGRAPH 85*, pages 31–40, San Francisco, July 1985.

[22] R. L. Cook and K. E. Torrance. A reflectance model for computer graphics. *ACM Comput. Graphics*, 15(3):307–316, 1981.

[23] S.W. Draper. The use of gradient and dual space in line-drawing interpretation. *Artificial Intelligence*, 17:461–508, 1981.

[24] D. Eggert and K. Bowyer. Perspective projection aspect graphs of solids of revolution: An implementation. In *IEEE Workshop on Directions in Automated CAD-Based Vision*, pages 44–53, June 1991.

[25] D. Eggert, K. Bowyer, C. Dyer, H. Christensen, and D. Goldgof. The scale space aspect graph. In *Proc. IEEE Conf. Comp. Vision Patt. Recog.*, pages 335–340, 1992.

[26] G. Falk. Interpretation of imperfect line data as a three-dimensional scene. *Artificial Intelligence*, 3:101–144, 1972.

[27] R.T. Farouki. The characterization of parametric surface sections. *Comp. Vis. Graph. Im. Proc.*, 33:209–236, 1986.

[28] O.D. Faugeras and M. Hebert. The representation, recognition, and locating of 3-D objects. *International Journal of Robotics Research*, 5(3):27–52, Fall 1986.

[29] B. Faverjon and J. Ponce. On computing two-finger force-closure grasps of curved 2D objects. In *IEEE Int. Conf. on Robotics and Automation*, pages 424–429, Sacramento, CA, April 1991.

[30] P.J. Flynn. 3-d object recognition with symmetric models: Symmetry extraction and encoding. *IEEE Trans. Patt. Anal. Mach. Intell.*, 16:814–818, 1994.

[31] D. Forsyth and A. Zisserman. Reflections on shading. *IEEE Trans. Patt. Anal. Mach. Intell.*, 13(7):671–679, July 1991.

[32] T. L. Gandhi and O. Camps. Robust feature selection for object recognition using uncertain 2D image data. In *Proc. IEEE Conf. Comp. Vision Patt. Recog.*, pages 281–287, Seattle, Washington, June 1994.

[33] P.C. Gaston and T. Lozano-Pérez. Tactile recognition and localization using object models: The case of polyhedra in the plane. *IEEE Trans. Patt. Anal. Mach. Intell.*, 6(3), 1984.

[34] Z. Gigus, J. Canny, and R. Seidel. Efficiently computing and representing aspect graphs of polyhedral objects. *IEEE Trans. Patt. Anal. Mach. Intell.*, 13(6), June 1991.

[35] Z. Gigus and J. Malik. Computing the aspect graph for line drawings of polyhedral objects. *IEEE Trans. Patt. Anal. Mach. Intell.*, 12(2):113–122, February 1990.

[36] W. E. L. Grimson and D. P. Huttenlocher. On the verification of hypothesized matches in model-based recognition. *IEEE Trans. Patt. Anal. Mach. Intell.*, 13(12):1201–1213, December 1991.

[37] W. E. L. Grimson, D. P. Huttenlocher, and T. D. Alter. Recognizing 3D objects from 2D images; an error analysis. In *Proc. IEEE Conf. Comp. Vision Patt. Recog.*, pages 316–321, Champaign, Illinois, June 1992.

[38] W.E.L. Grimson and T. Lozano-Pérez. Model-based recognition and localization from sparse range or tactile data. *International Journal of Robotics Research*, 3(3), 1984.

[39] A. Guzman. Computer recognition of three-dimensional objects in a visual scene. Technical Report MAC-TR-59, MIT, 1968.

[40] R. M. Haralick, C. Lee, K. Ottenberg, and M. Nölle. Surface reflection: Physical and geometrical perspectives. *Int. J. of Comp. Vision*, 13(3):331–356, December 1994.

[41] G. Healey and T.O. Binford. Local shape from specularity. In *Proc. DARPA Image Understanding Workshop*, pages 874–877, 1987.

[42] B.K.P Horn. Obtaining shape from shading information. In P.H. Winston, editor, *The Psychology of Computer Vision*, pages 115–155. McGraw-Hill, New York, 1975.

[43] B.K.P. Horn. *Computer Vision.* MIT Press, Cambridge, Mass., 1986.

[44] D.A. Huffman. Impossible objects as nonsense sentences. *Machine Intelligence*, 6:295–323, 1971.

[45] D.A. Huffman. Realizable configurations of lines in pictures of polyhedra. *Machine Intelligence*, 8:493–509, 1977.

[46] D. Huttenlocher and S. Ullman. Recognizing 3D solid objects by alignment with an image. *Int. J. of Comp. Vision*, 5(2):195–212, 1990.

[47] K. Ikeuchi and B.K.P. Horn. Numerical shape from shading and occluding boundaries. *Artificial Intelligence*, 17:141–184, 1981.

[48] D. Kahaner, C. Moler, and S. Nash. *Numerical Methods and Software.* Prentice Hall, Englewood Cliffs, NJ, 1989.

[49] T. Kanade. A theory of the Origami world. *Artificial Intelligence*, 13:279–311, 1980.

[50] T. Kanade. Recovery of the three-dimensional shape of an object from a single view. *Artificial Intelligence*, 17:409–460, 1981.

[51] J. Kender and D. Freudenstein. What is a degenerate view. In *Proc. International Joint Conference on Artificial Intelligence*, pages 801–804, August 1987.

[52] Y.L. Kergosien. Generic sign systems in medical imaging. *IEEE Computer Graphics and Applications*, 11(5):46–65, 1991.

[53] R. Kimmel and A. Bruckstein. Global shape from shading. *Comp. Vis. Graph. Im. Understanding*, 1995. to appear.

[54] J.J. Koenderink and A.J. Van Doorn. The internal representation of solid shape with respect to vision. *Biological Cybernetics*, 32:211–216, 1979.

[55] D.J. Kriegman and J. Ponce. A new curve tracing algorithm and some applications. In P.J. Laurent, A. Le Méhauté, and L.L. Schumaker, editors, *Curves and Surfaces*, pages 267–270. Academic Press, New York, 1991.

[56] D. Lowe. *Perceptual Organization and Visual Recognition.* Kluwer Academic Publishers, Boston, 1985.

[57] D.G. Luenberger. *Linear and nonlinear programming.* Addison-Wesley, 1984. Second edition.

[58] A.K. Mackworth. Interpreting pictures of polyhedral scenes. *Artificial Intelligence*, 4(2):121–137, June 1973.

[59] G. W. Meyer, H. E. Rushmeier, M. F. Cohen, D. P. Greenberg, and K. E. Torrance. An experimental evaluation of computer graphics imagery. *ACM Transactions on Graphics*, 5(1):30–50, 1986.

[60] J.J. Moré, B.S. Garbow, and K.E. Hillstrom. User guide for MINPACK-1. ANL-80-74, Argonne National Laboratories, 1980.

[61] A.P. Morgan. *Solving Polynomial Systems using Continuation for Engineering and Scientific Problems.* Prentice Hall, Englewood Cliffs, NJ, 1987.

[62] S. K. Nayar, K. Ikeuchi, and T. Kanade. Shape from interreflections. *Int. J. of Comp. Vision*, 6(3):173–195, August 1991.

[63] S. K. Nayar, K. Ikeuchi, and T. Kanade. Surface reflection: Physical and geometrical perspectives. *IEEE Trans. Patt. Anal. Mach. Intell.*, 13(7):611–634, July 1991.

[64] S.K. Nayar and M. Oren. Diffuse reflectance from rough surfaces. In *Proc. IEEE Conf. Comp. Vision Patt. Recog.*, pages 763–764, 1993.

[65] V.D. Nguyen. Labeling polyhedral scenes. In *Proc. DARPA Image Understanding Workshop*, pages 1160–1165, April 1988.

[66] C. F. Olsen. Fast alignment using probabilistic indexing. In *Proc. IEEE Conf. Comp. Vision Patt. Recog.*, pages 387–392, New York, New York, June 1993.

[67] C. F. Olsen. Probabilistic indexing for object recognition. *IEEE Trans. Patt. Anal. Mach. Intell.*, 17(5):518–521, May 1995.

[68] S. Petitjean. *Géométrie énumérative et contacts de variétés linéaires: application aux graphes d'aspects d'objects courbes.* PhD thesis, Institut National Polytechnique de Lorraine, 1995.

[69] S. Petitjean, J. Ponce, and D.J. Kriegman. Computing exact aspect graphs of curved objects: Algebraic surfaces. *Int. J. of Comp. Vision*, 9(3), 1992.

[70] H. Plantinga and C. Dyer. Visibility, occlusion, and the aspect graph. *Int. J. of Comp. Vision*, 5(2):137–160, 1990.

[71] J. Ponce and I. Shimshoni. An algebraic approach to line-drawing analysis in the presence of uncertainty. In *IEEE Int. Conf. on Robotics and Automation*, pages 1786–1791, 1992.

[72] F. P. Preparata and M.I. Shamos. *Computational Geometry, An Introduction.* Springer-Verlag, New York, 1985.

[73] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes in C.* Cambridge University Press, 1988.

[74] J.H. Rieger. Global bifurcations sets and stable projections of non-singular algebraic surfaces. *Int. J. of Comp. Vision*, 7(3):171–194, 1992.

[75] J.H. Rieger. On the complexity and computation of view graphs of piecewise-smooth algebraic surfaces. Technical Report FBI-HH-M-228/93, Universit at Hamburg, 1993.

[76] W.B. Seales and C.R. Dyer. Constrained viewpoint from occluding contour. In *IEEE Workshop on Directions in Automated "CAD-Based" Vision*, pages 54–63, Maui, Hawaii, June 1991.

[77] R. Shapira. A note on Sugihara's claim. *IEEE Trans. Patt. Anal. Mach. Intell.*, 6(1):122–123, January 1984.

[78] I. Shimshoni and J. Ponce. Finite resolution aspect graphs of polyhedral objects. In *IEEE Workshop on Qualitative Vision*, pages 140–150, New York, New York, June 1993.

[79] I. Shimshoni and J. Ponce. Recovering the shape of polyhedra using line-drawing analysis and complex models. In *Proc. IEEE Conf. Comp. Vision Patt. Recog.*, pages 514–519, Seattle, Washington, June 1994.

[80] I. Shimshoni and J. Ponce. Probabilistic 3D object recognition. In *Proc. Int. Conf. Comp. Vision*, pages 488–493, Boston, Massachusetts, June 1995.

[81] R. Siegal and J. R. Howell. *Thermal Radiation Heat Transfer*. Hemisphere Publishing, New York, 1992.

[82] T. Sripradisvarakul and R. Jain. Generating aspect graphs of curved objects. In *Proc. IEEE Workshop on Interpretation of 3D Scenes*, pages 109–115, Austin, TX, December 1989.

[83] J. Stewman and K.W. Bowyer. Aspect graphs for planar-face convex objects. In *Proc. IEEE Workshop on Computer Vision*, pages 123–130, Miami, FL, 1987.

[84] J. Stewman and K.W. Bowyer. Creating the perspective projection aspect graph of polyhedral objects. In *Proc. Int. Conf. Comp. Vision*, pages 495–500, Tampa, FL, 1988.

[85] K. Sugihara. An algebraic approach to the shape-from-image-problem. *Artificial Intelligence*, 23:59–95, 1984.

[86] K. Sugihara. A necessary and sufficient condition for a picture to represent a polyhedral scene. *IEEE Trans. Patt. Anal. Mach. Intell.*, 6(5):578–586, September 1984.

[87] H. D. Tagare and Rui J. P. deFigueriedo. Simultaneous estimation of shape and reflectance maps from photometric stereo. In *Proc. Int. Conf. Comp. Vision*, pages 223–230, Boston, Massachusetts, December 1988.

[88] K.E. Torrance and E.M. Sparrow. Theory for off-specular reflections from roughened surfaces. *Journal of the Optical Society of America*, 57:1105–1114, September 1967.

[89] D. L. Waltz. Understanding line drawings of scenes with shadows. In P.H. Winston, editor, *The Psychology of Computer Vision*, pages 19–91. McGraw-Hill, New York, 1975.

[90] R. Wang and H. Freeman. Object recognition based on characteristic views. In *International Conference on Pattern Recognition*, pages 8–12, Atlantic City, NJ, June 1990.

[91] N. Watts. Calculating the principal views of a polyhedron. CS Tech. Report 234, Rochester University, 1987.

[92] D. Weinshall, M. Werman, and N. Tishby. Stability and likelihood of views of three dimensional objects. In *Proc. European Conf. Comp. Vision*, pages 24–35, Stockholdm, Sweden, June 1994.

[93] M. D. Wheeler and K. Ikeuchi. Sensor modeling, probabilistic hypothesis generation, and robust localization for object recognition. *IEEE Trans. Patt. Anal. Mach. Intell.*, 17(3):252–265, March 1995.

[94] R. J. Woodham. Photometric stereo: A reflectance map technique for determining surface orientation from image intensity. *In Proc. SPIE*, 155:136–143, 1978.

[95] Y. Wu, S. S. Iyenger, R. Jain, and S. Bose. A new generalized computational framework for finding object orientation using perspective trihedral angle constraint. *IEEE Trans. Patt. Anal. Mach. Intell.*, 16(10):961–975, October 1994.

Curriculum Vitae

Ilan M. Shimshoni

ADDRESS

The Beckman Institute
405 North Mathews Avenue
Urbana, IL 61801

Phone: (217) 244-1372
E-mail: ilan@cs.uiuc.edu

Born: August Third 1959, Rehovot Israel

EDUCATION

Ph.D. "Interpreting Images of Polyhedral Objects in the Presence of Uncertainty", Department of Computer Science, the University of Illinois at Urbana-Champaign. Advisor: Professor Jean Ponce.

M.S. "Time/space tradeoff algorithms for the permutation representation problem", Weizmann Institute of Science, Rehovot, Israel, December 1989. Advisor: Professor Adi Shamir.

B.S. Hebrew University, Jerusalem, Israel, June 1984.

EMPLOYMENT

1991-present: Research Assistant, Department of Computer Science, University of Illinois.

1984-1991: Software Engineer, Adcad Ltd, Rehovot, Israel and i-Logix Inc, Boston, Ma. In charge of software design of the "Statemate" software engineering tool for specification of reactive systems.

1985-1986: Research Assistant, Robotics Laboratory, Weizmann Institute of Science.

1977-1982: Computer Programmer, Israel Defense Forces.

RESEARCH INTERESTS

- Probabilistic Reasoning in Computer Vision
- Line-Drawing Analysis
- Robot Motion Planning
- Physics-Based Vision
- Aspect Graphs

PROFESSIONAL ACTIVITIES

Reviewer: IEEE Trans. on Pattern Analysis and Machine Intelligence; Computer Vision Graphics and Image Processing; IEEE Computer Vision and Pattern Recognition Conference; International Conference on Pattern Recognition; International Conference on Computer Vision.

Member: IEEE.

HONORS

Member of Phi Kappa Phi, Chapter of The University of Illinois at Urbana-Champaign.

Graduated with Honors 1984, Hebrew University, Jerusalem.

PUBLICATIONS

1. I. Shimshoni R. Kimmel and A. Bruckstein, "The Completeness of The Global Shape from Shading Algorithm," Under review *Comp. Vision Image Understanding.*

2. I. Shimshoni and J. Ponce, "3D Probabilistic Object Recognition," To appear in *Int. Conf. on Computer Vision*, June 1995.

3. I. Shimshoni and J. Ponce, "Recovering the shape of polyhedra using line-drawing analysis and complex reflectance models," In *Proc. IEEE Conf. Comp. Vision Patt. Recog.*, Seattle, Washington, June 1994. Also Beckman Institute Technical Report UIUC-BI-AI-RCV-93-09, University of Illinois at Urbana-Champaign.

4. I. Shimshoni and J. Ponce, "Finite resolution aspect graphs of polyhedral objects," In *IEEE Workshop on Qualitative Vision*, pages 140–150, New York, New York, June 1993. Also in *Israeli Symposium on Artificial Intelligence Computer Vision and Neural Networks*, pages 505–514, Ramat Gan, Israel, December 1993.

5. J. Ponce and I. Shimshoni, "An algebraic approach to line-drawing analysis in the presence of uncertainty," In *IEEE Int. Conf. on Robotics and Automation*, pages 1786–1791, Nice, France, May 1992.

6. A. Fiat, S. Moses, A. Shamir, I. Shimshoni, and G. Tardos, "Planning and learning in permutation groups," In *Proc. of the 30th Ann. IEEE Symp. on Foundations of Computer Science*, pages 274–279, 1989.