



This is to certify that the

dissertation entitled

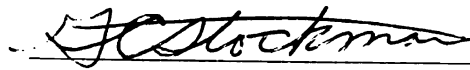
RECONSTRUCTION OF LINE DRAWING
GRAPHS FROM FUSED RANGE AND
INTENSITY IMAGERY

presented by

GREG CHUNGMOU LEE

has been accepted towards fulfillment
of the requirements for

PhD degree in Computer Science


Major professor

Date 6 Aug 92

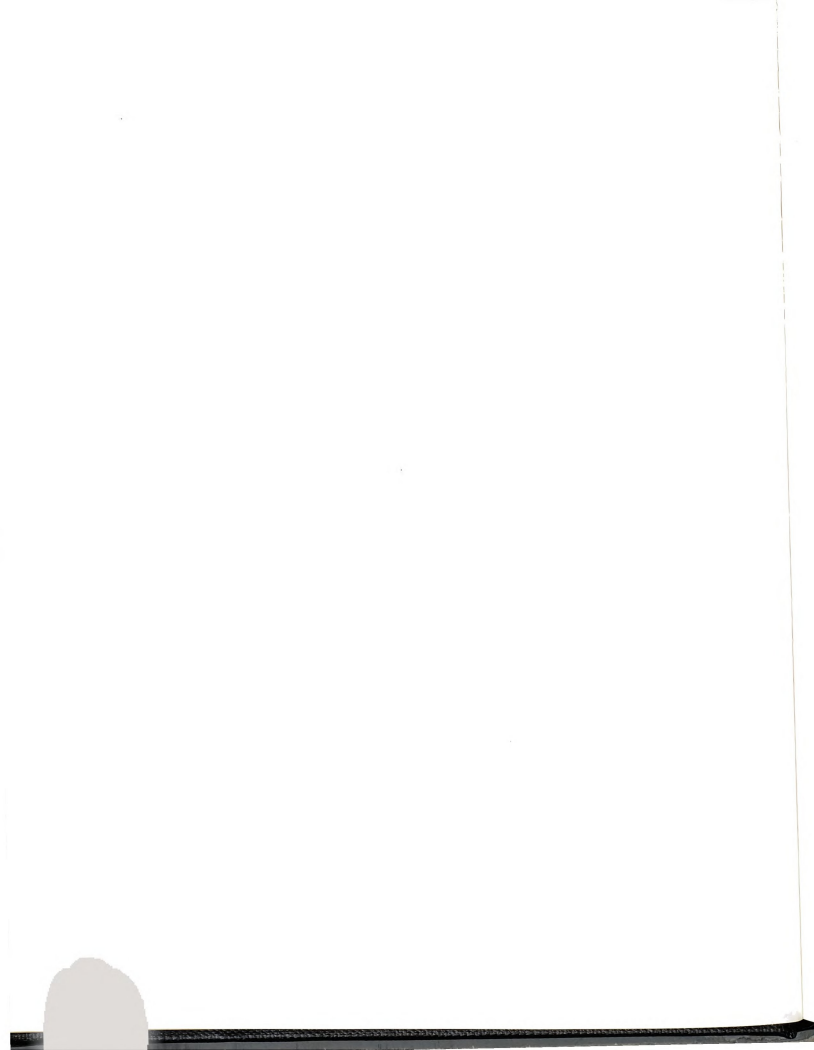
LIBRARY
Michigan State
University

PLACE IN RETURN BOX to remove this checkout from your record.
TO AVOID FINES return on or before date due.

DATE DUE	DATE DUE	DATE DUE
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____

MSU Is An Affirmative Action/Equal Opportunity Institution

c:\crl\data\due.pm:3-p.



RECONSTRUCTION OF LINE DRAWING
GRAPHS FROM FUSED RANGE AND
INTENSITY IMAGERY

By

Greg Chungmou Lee

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Computer Science Department

1992

ABSTRACT

RECONSTRUCTION OF LINE DRAWING GRAPHS FROM FUSED RANGE AND INTENSITY IMAGERY

By

Greg Chungmou Lee

This thesis addresses the problem of extracting a labeled line drawing graph from registered range and intensity imagery. *Wing* representation [26] is used as an intermediate step to achieve the final goal.

To attain the wing representation of an imaged scene, a low level feature detection algorithm for extracting wing primitives is proposed. The procedure is largely based on a hypothesize-and-test strategy. In a small neighborhood of the fused image, existence of various wing primitives is hypothesized. Verification of a wing hypothesis is posed as a non-linear optimization problem in which the 2D intensity contour constraint, whenever available, is used to aid the recovery of surface parameters. Fitting functions that relate various quadric surface forms to their limb contours and intersecting quadric surfaces to the projection of the crease edges are derived to define the constraints of fused data fitting. The limb contours of quadric surfaces are also instrumental in deducing good initial parameter estimates for the non-linear optimization

scheme. Results from Monte Carlo studies on the accuracy of parameter estimates and surface shape discrimination by fitting fused, surface-only and contour-only data confirm the importance of the limb contour constraints. Further study shows that only very few data points are needed to achieve quality results. Experimental results of wing detection on real images indicate that a recognition system probably can be built upon the outlined primitive detection subsystem.

Given the wing representation of the scene, the labeled line drawing graph representation is constructed. Three origami and polygonal scene reconstruction procedures are given to analyze scenes under various restrictive assumptions. With very idealistic assumptions about the wing samples, the complete and unique LLDG is tenable if resampling of the range image is allowed. By studying the geometric properties of the line drawing graphs, a set of necessary but not sufficient rules are derived to reconstruct the LLDG without having to resample the range image; however, uniqueness of the reconstructed LLDG is not guaranteed. In dealing with imperfect wing sampling, a heuristic based algorithm is devised to handle the possibility of having missing and spurious wings and the inherent measurement and computation errors. Methods for reconstructing quadric surface scenes are proposed but not implemented.

All of the implemented algorithms have been tested on both synthetic and real images. Parallel renditions are natural but not yet attempted. It is shown that reconstruction of labeled line drawing graphs from raw fused images via wing features is plausible.

Copyright © by
Greg Chungmou Lee
1992



To my parents
and my wife, Hsin-Chen

I wish to thank toward the successful patience helped keep countless hours at mtered in research. I personally. Acknowledgers Anil Jain, Mihir suggestions helped n and maintaining the with first class equip Professor Richard Du Monte Carlo experin Barton for giving me The experience was b

The alliance of thabar, and I, came ab all started on thesis v one week of each othe up the spirit when on the completion of this

I am grateful for th former and current PI Sei-Wang Chen for int and Dr. Patrick Flynn providing many useful value the advice of Dr. thesis preparation.

I also wish to than tachejee, Jinlong Ch

ACKNOWLEDGMENTS

I wish to thank my thesis advisor, Professor George Stockman, for his guidance toward the successful completion of this thesis. His directions, encouragements, and patience helped keep me focused and motivated. I sincerely appreciate his spending countless hours at nights and on weekends to analyze the problems I have encountered in research. He has influenced me in a positive way both academically and personally. Acknowledgment is also due to my thesis committee members, Professors Anil Jain, Mihran Tuceryan, Wen-Jing Hsu and Connie Page, whose invaluable suggestions helped make a better thesis. Special thanks to Dr. Jain for equipping and maintaining the Pattern Recognition and Image Processing (PRIP) laboratory with first class equipment for conducting research. I value the critical comments of Professor Richard Dubes during my PRIP presentations and his advice on conducting Monte Carlo experiments. I also wish to thank Professors Richard Reid and Ruth Barton for giving me the opportunity to participate on the ACM Programming team. The experience was both interesting and worthwhile.

The alliance of the gang-of-four, Deborah Trytten, Narayan Raja, Sateesha Nadabar, and I, came about when we took the comprehensive examination together. We all started on thesis writing at about the same time and defended our theses within one week of each other. We shared the joy of successful career searches and cheered up the spirit when one of us was down. Their emotional support was instrumental on the completion of this thesis. To them, I say thank you, my friends.

I am grateful for the friendship and the thought provoking discussions with many former and current PRIPies. I especially wish to thank Drs. Chaur-Chin Chen and Sei-Wang Chen for introducing me to the wonderful research area of Computer Vision and Dr. Patrick Flynn for paving the way for future research using range image by providing many useful software tools for obtaining and processing range images. I value the advice of Dr. Joseph Miller and Dr. Farshid Farrokhnia on research and on thesis preparation.

I also wish to thank all other PRIPies, especially Philippe Ballard, Sushil Bhat-tarjee, Jinlong Chen, Yao Chen, John Courtney, Chittra Dorai, Marie-Pierre

Dubuisson, Hans
Nanda, Tim Newm
tenure an enjoyabl
four to seize contro
grateful to Steve W
finish up his resear
Les and former ma
throughout the yea

This research h
grant CDA-8806599
College of Engineeri
of Michigan State U
Department of Corr
throughout my grad
conferences held aro

Finally, I must p
life. My most since
through my beginnin
current success. I ac
education, of my fam
wife, Hsin-Chen, for

Dubuisson, Hans Dulimarta, Sally Howden, Qian Huang, Jian-Chang Mao, Arun Nanda, Tim Newman, Philippe Ohanian, and Sharathcha Pankanti, for making my tenure an enjoyable one. I value their kinship and support for allowing the gang-of-four to seize control of the PRIP lab during the thesis writing stage. I am especially grateful to Steve Walsh for proofreading this thesis at a time when he is also trying to finish up his research. My sincere appreciation also goes to the PRIP manager, John Lees and former manager, Dave Marks, for maintaining a stable research environment throughout the years.

This research has been supported in part by the National Science Foundation grant CDA-8806599 and IRI 8903628, by the Graduate School Fellowship, by the College of Engineering Research Fellowship and by the Equal Opportunity Fellowship of Michigan State University. Special thanks to Professor Anthony Wojcik and the Department of Computer Science for supporting me with a Graduate Assistantship throughout my graduate studies and for the travel funds that enabled me to attend conferences held around the country.

Finally, I must pay tributes to the people whom have made a big difference in my life. My most sincere gratitude to Mr. and Mrs. Hugh Fosters for their guidance through my beginning years in the United States. They are most instrumental to my current success. I acknowledge the love of my parents for providing me with the best education, of my family for their encouragements along the way, and especially of my wife, Hsin-Chen, for her understanding and support through my education endeavor.

LIST OF TABLES

LIST OF FIGURES

1 Introduction

1.1 Problem Definition

1.1.1 Wireframe Models

1.2 Representation

1.3 Organization

2 Formalization

2.1 Introduction

2.2 Object Domains

2.2.1 δ -polyhedra

2.2.2 δ -quadrilaterals

2.2.3 Sampling

2.3 Accidental Intersections

2.4 The Line Drawing

2.5 What are Wireframe Models?

2.5.1 Wing Models

2.5.2 Derivatives

2.5.3 Derivatives

2.6 Registered Features

2.7 Summary

3 Wing Detection

3.1 Introduction

3.2 Survey of Related Work

3.2.1 Edge Detection

3.2.2 Segmentation

3.2.3 3D Surface Reconstruction

TABLE OF CONTENTS

LIST OF TABLES	xii
LIST OF FIGURES	xiii
1 Introduction	1
1.1 Problem Definition	3
1.1.1 Wing Representation Theory	4
1.2 Representational Schemes for 3D Object Recognition	7
1.3 Organization of the Dissertation	11
2 Formalization of Object Domain and Wing Models	14
2.1 Introduction	14
2.2 Object Domain Definitions	15
2.2.1 δ -polygons	16
2.2.2 δ -quadric surfaces	16
2.2.3 Sample of Objects in the Object Domain	19
2.3 Accidental and Non-Accidental Viewpoints	21
2.4 The Line Drawing Graph (LDG)	23
2.5 What are Wings?	24
2.5.1 Wing Primitives	25
2.5.2 Derivation of Wing Contour Equations	27
2.5.3 Deriving P_{2d} from P_{3d}	28
2.6 Registered Fused Images	34
2.7 Summary	35
3 Wing Detection	37
3.1 Introduction	37
3.2 Survey of Related Background	38
3.2.1 Edge Detection in Intensity Imagery	39
3.2.2 Segmentation of Range Imagery	40
3.2.3 3D Surface Reconstruction: Superquadric and Quadric Fitting	41

3.3	Wing Det
3.3.1	Te
3.3.2	De
3.3.3	Fi
3.4	Simultane
3.4.1	Fi
3.4.2	Di
3.4.3	χ^2
3.4.4	Ini
3.4.5	An
3.5	Experimen
3.5.1	Fitt
3.5.2	Sun
3.5.3	Sim
3.6	Fitting wit
3.7	Experimen
3.7.1	JIG
3.7.2	CY
3.7.3	Ope
3.7.4	Sun
3.7.5	Mon
3.8	Summary

4	Perfect Recon
4.1	Introduction
4.2	Survey of R
4.2.1	Mod
4.2.2	Mod
	Orig
4.2.3	Mod
	World
4.3	Reconstructi
4.3.1	Assu
4.3.2	Algor
4.3.3	Resar
4.4	Reconstructi
4.4.1	New
4.4.2	Geom

3.3	Wing Detection Algorithm	43
3.3.1	Tessellation of the Input Image	44
3.3.2	Detecting Presence of Wings	45
3.3.3	Fitting Wing Primitives	48
3.4	Simultaneous Boundary and Surface Fitting	49
3.4.1	Fitting Equations	51
3.4.2	Distance Functions	52
3.4.3	χ^2 Merit Functions	54
3.4.4	Initial Parameters	56
3.4.5	An Example	61
3.5	Experiments with Synthetic Data	63
3.5.1	Fitting True Models to Contour, Surface, and Fused Data . .	64
3.5.2	Surface Classification Experiments	72
3.5.3	Simulation Results Summary	78
3.6	Fitting with Different Number of Surface and Contour Points	79
3.7	Experiments with Real Fused Imagery	80
3.7.1	JIG1 Example	80
3.7.2	CYL2 Example	81
3.7.3	Open Cup vs. Lid Cup	85
3.7.4	Summary on 10 Real Images	87
3.7.5	More Wing Detection Examples	89
3.8	Summary	91
4	Perfect Reconstruction of LLDG - Origami/Polyhedral World	93
4.1	Introduction	93
4.2	Survey of Related Background	95
4.2.1	Model Based Interpretation of Line Drawings	96
4.2.2	Model-Free Interpretation of Line Drawings - Polyhedral and Origami World	96
4.2.3	Model-Free Interpretation of Line Drawings - Curved Surface World	102
4.3	Reconstruction of LDG Via Resampling	103
4.3.1	Assumptions	105
4.3.2	Algorithmic Approach	105
4.3.3	Resampling Reconstruction Algorithm: POLY-1	110
4.4	Reconstruction Via Geometric Constraints	111
4.4.1	New Terminology	112
4.4.2	Geometric Constraints Imposed by Projection of a δ -polygon	113

4.4.3 Ge

4.4.4 Co

4.4.5 Ex

4.4.6 Me

4.5 Experimen

4.6 Worst Cas

4.7 Summary

5 Partial Recon

5.1 Introduction

5.2 Survey of

5.3 Heuristic I

5.3.1 Pre

5.3.2 Clu

5.3.3 Not

5.3.4 Heu

5.3.5 Men

5.3.6 Heu

5.4 Examples

5.5 Worst Case

5.6 Summary

6 Reconstruction

6.1 Introduction

6.2 Perfect Rec

6.2.1 Win

6.2.2 Win

6.2.3 Exte

6.2.4 Reco

6.2.5 Quac

6.3 Heuristic Re

6.3.1 Wing

6.3.2 Wing

6.3.3 A He

6.3.4 Heur

6.3.5 Exam

6.4 Summary .

4.4.3	Geometric Constraint Based Decision Rules	117
4.4.4	Complete Reconstruction Algorithm: POLY-2	123
4.4.5	Example - Paddle Wheel	124
4.4.6	Merging of Surfaces	125
4.5	Experimental Results	131
4.6	Worst Case Time Complexities of POLY-1 and POLY-2	132
4.7	Summary	140
5	Partial Reconstruction of LLDG - Origami/Polyhedral World	142
5.1	Introduction	142
5.2	Survey of Related Background	143
5.3	Heuristic LLDG Reconstruction Algorithm	146
5.3.1	Preprocessing of Wing Samples	147
5.3.2	Clustering of Wings	148
5.3.3	Notion of Maybe-Segments	149
5.3.4	Heuristic Rules	149
5.3.5	Merging of Planar Faces	155
5.3.6	Heuristic Reconstruction Algorithm: POLY-3	158
5.4	Examples	159
5.5	Worst Case Time Complexity of POLY-3	165
5.6	Summary	169
6	Reconstruction of the LLDG - Quadric Surface Scenes	172
6.1	Introduction	172
6.2	Perfect Reconstruction Algorithm	173
6.2.1	Wing Sensing Assumptions	173
6.2.2	Wing Clustering	174
6.2.3	Extension of Wings	174
6.2.4	Reconstruction of Individual Wing Group	175
6.2.5	Quadric LLDG Reconstruction Algorithm: QUAD-1	177
6.3	Heuristic Reconstruction Algorithm	180
6.3.1	Wing Clustering	180
6.3.2	Wing Extension	181
6.3.3	A Heuristic Approach	182
6.3.4	Heuristic Quadric LLDG Reconstruction Algorithm: QUAD-2	189
6.3.5	Examples	189
6.4	Summary	190

7 Summary, Conclusions

7.1 Summary

7.2 Contributions

7.3 Recommendations

A Creation of Functions

B χ^2 Merit Function

C Parameters of Algorithms

BIBLIOGRAPHY

7	Summary, Conclusions and Recommendations for Future Research	194
7.1	Summary and Conclusions	194
7.2	Contributions	198
7.3	Recommendations for Future Research	200
A	Creation of Fused Imagery	203
B	χ^2 Merit Function for Fused Fitting	214
C	Parameters of the Wing Detection and the LLDG Reconstruction Algorithms	216
	BIBLIOGRAPHY	221

- 2.1 List of all
- 2.2 List of 38
- 3.1 Range (in
- 3.2 Initial esti
- 3.3 Fitting Sp
- 3.4 Fitting Cy
- 3.5 Fitting Co
- 3.6 Fitting Pla
- 3.7 Fitting all
- 3.8 Fitting all
- 3.9 Fitting all
- 3.10 Fitting all
- 3.11 Percentage
- 3.12 Surface-onl
- 3.13 Summary o
- 4.1 Rest of the
- 5.1 Summary o
- C.1 Parameters
- C.2 Surface Moc
- C.3 Parameters

LIST OF TABLES

2.1	List of all 60 half-wing primitives	27
2.2	List of 38 full wing primitives	28
3.1	Range (in inches) of parameters	64
3.2	Initial estimate of the model parameters	65
3.3	Fitting Spherical Model on Spherical Data	66
3.4	Fitting Cylindrical Model on Cylindrical Data	68
3.5	Fitting Conical Model on Conical Data	70
3.6	Fitting Planar Model on Planar Data	71
3.7	Fitting all models to Spherical surface patch	73
3.8	Fitting all models to Cylindrical surface patch	75
3.9	Fitting all models to Conical surface patch	76
3.10	Fitting all models to Planar surface patch	78
3.11	Percentage of bad fused fits versus number of surface and contour points.	79
3.12	Surface-only fitting with various numbers of surface points.	80
3.13	Summary of wing detection of 10 real images	88
4.1	Rest of the processing steps of the Paddle Wheel example.	125
5.1	Summary of the LLDG of the 7 real images.	166
C.1	Parameters of the wing detection algorithm	217
C.2	Surface Model parameters	219
C.3	Parameters of the POLY-3 algorithm	220

- 1.1 High level
reconstruct
- 1.2 Unique win
which all h
- 1.3 Aspect gra
- 2.1 Examples c
- 2.2 Examples c
- 2.3 (a) Exampl
not in the c
- 2.4 (a) A non-a
under [83, 9
not be an a
from the bo
- 2.5 Representat
junction cat.
resentation.
- 2.6 Setup of Wh
- 3.1 Flowchart of
- 3.2 Tessellation c
- 3.3 Wing detecti
- 3.4 Data points
errors. . . .
- 3.5 Computing t
of a cone. . .
- 3.6 Fitting wing
- 3.7 Detected wing
- 3.8 Detected wing

LIST OF FIGURES

1.1	High level description and an example of wing detection and LLDG reconstruction process	2
1.2	Unique wing representation of a bowl, a half grapefruit and a clam, which all have the same line drawing [27]	5
1.3	Aspect graph of a tetrahedron (from [71]).	11
2.1	Examples of valid and invalid δ -polygons	17
2.2	Examples of valid and invalid δ -quadrics	19
2.3	(a) Examples of objects in the object domain. (b) Examples of objects not in the object domain.	20
2.4	(a) A non-accidental view of a solid box, which is considered accidental under [83, 91] (b) An accidental view of a topless box. (This would not be an accidental view if the side containing a , b , c were removed from the box!)	22
2.5	Representation of a coke can. (a) Line labeling using Malik's [83] junction catalogue (b) Chen's wing [27] representation. (c) LLDG representation.	25
2.6	Setup of White Scanner and the two coordinate frames	35
3.1	Flowchart of proposed wing detection algorithm	44
3.2	Tessellation of input image	45
3.3	Wing detection within each subimage w_i	47
3.4	Data points that are close to the fitted curve may have large residual errors.	50
3.5	Computing the Euclidean distance from a point in 3D to the surface of a cone.	55
3.6	Fitting wing models to a window with cylindrical surface patch.	62
3.7	Detected wings of the JIG image	82
3.8	Detected wings of the CYL image	83

3.9 Wing repr
lid . . .

3.10 Wing dete

4.1 Wing repr

4.2 Huffman-C

4.3 Problems

4.4 Origami ju

type (from

4.5 Junction c

4.6 Apiece of r

4.7 Deciding o

4.8 Graphical

4.9 Graphical

4.10 Multiple re

4.11 Graphical i

4.12 Graphical i

4.13 Graphical i

4.14 Reconstruct

4.15 Reconstruct

4.16 Applying A

4.17 Reconstruct

4.18 Complete re

4.19 More exam

4.20 Reconstruct

5.1 Falk's heuris

5.2 Spurious line

5.3 Some exam

5.4 Graphical ill

5.5 Problems en

5.6 Reconstructi

5.7 Examples of

6.1 Examples of

scenes. . . .

6.2 Determining

6.3 Graphical illu

curved object

3.9	Wing representation can distinguish an opened cup from a cup with a lid	86
3.10	Wing detection examples of quadric surface scenes	90
4.1	Wing representation and reconstruction of the Big-Block (from [26]).	94
4.2	Huffman-Clowes junction catalogue.	98
4.3	Problems with Huffman-Clowes labeling scheme	99
4.4	Origami junction types and number of possible junction labels for each type (from [66]).	100
4.5	Junction catalogue for piecewise smooth surfaces [83]	104
4.6	Apiece of nose cone is not in Malik's [83] object domain.	104
4.7	Deciding on LDG line segments (Figure originated from [26]).	108
4.8	Graphical interpretation of newly defined functions.	113
4.9	Graphical illustration of Theorem 4.3	116
4.10	Multiple region labels are possible if wings are collinear.	118
4.11	Graphical illustration of the redefined and the new functions	120
4.12	Graphical illustration of Rule 4.3	122
4.13	Graphical illustration of Rule 4.4	123
4.14	Reconstruction of the background wing group of the Paddle Wheel.	128
4.15	Reconstructed silhouette (background) of the Paddle Wheel.	129
4.16	Applying Algorithm 2 on front face of the Big Block	130
4.17	Reconstructed LDG of the Big Block.	131
4.18	Complete reconstruction of the Big Block	133
4.19	More examples of syntactic scenes that have been reconstructed	135
4.20	Reconstruction of the LLDG of a view of the gb3 block.	136
5.1	Falk's heuristic rule for completing imperfect line drawings [34]	145
5.2	Spurious line could be added with Falk's [34] rules.	146
5.3	Some examples of <i>Maybe</i> -segments and non- <i>Maybe</i> -segments	150
5.4	Graphical illustration of Heuristic Rules 2-6.	152
5.5	Problems encountered when merging faces into final LDG their solutions	157
5.6	Reconstruction of the JIG via algorithm POLY-2	161
5.7	Examples of reconstructed LLDG via algorithm POLY-2	164
6.1	Examples of reconstruction of LLDG from wing samples of quadric scenes.	179
6.2	Determining set of all possible junctions in the LDG	184
6.3	Graphical illustration of the heuristic rules for reconstructing LDG of curved object scenes	186

- 6.4 Reconstructing the original image
- 6.5 Reconstructing the original image using the LLL algorithm
- 6.6 More LLL

- A.1 Setup of the problem
- A.2 A complete set of basis vectors for the image, (c) image. . .
- A.3 Relating the basis vectors to the original image
To align the basis vectors with the original image about the new Y-axis
- A.4 Fused range

6.4	Reconstruction of LDG of a curved surface scene	187
6.5	Reconstruction of LLDGs from imperfect wing samples using heuristic algorithm QUAD-2.	191
6.6	More LLDGs that are reconstructed by the heuristic algorithm QUAD-2.	192
A.1	Setup of White Scanner and the two coordinate frames	205
A.2	A complete example: (a) original range image, (b) original intensity image, (c) calibration matrix, (d) fused range image, (e) fused intensity image.	207
A.3	Relating World coordinate frame to Camera-centered coordinate frame. To align the axes: rotate $\{C\}$ about X_c -axis by 45 degree then rotate about the new Z -axis by -90 degree follow by 180 rotation about the new Y -axis.	208
A.4	Fused range (left) and intensity (right) images	213

CHAPTER 1

Introduction

Human vision is so effortless that one often forgets that it is a very difficult task. Most people assume that the eye furnishes the brain with a description of the world in view. This is not so. The eye is just an input sensor; the visual cortex of the human brain is our primary organ of vision. An exact science of the human visual system is still unclear. Nevertheless, vision researchers continue to build computer vision systems to mimic human vision.

Computer vision has gone a long way since the days of Roberts [108]. The ultimate goal of a computer vision system is the recognition of the objects in the scene and the interpretation of their relationships. Thus it is not surprising that most research activities in the computer vision field concentrate on (1) building a better scene interpreter; (2) bettering the techniques of various object recognition modules; or (3) proposing altogether new object recognition and representation paradigms. Central to any object recognition system is the issue of object representation and feature extraction, which would be the function of the visual cortex in the human visual system. This thesis proposes new methods for extraction of *wing* features from raw input imagery and for construction of the labeled line drawing of the scene. An example of the goal of this work is depicted in Figure 1.1.

Figure 1.1. High level
construction process. 2
The middle image show
The bottom image show
discussed in Chapter 6

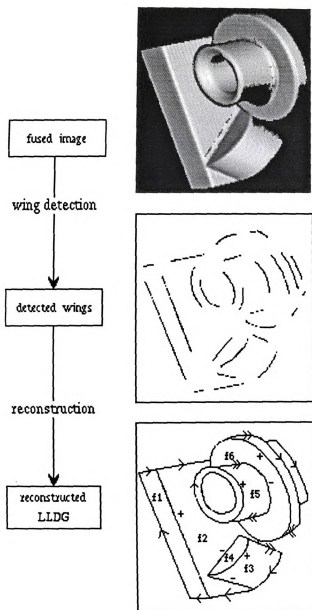


Figure 1.1. High level description and an example of wing detection and LLDG reconstruction process. The top image shows the intensity portion of the fused imagery. The middle image shows the ideal set of wings that are extracted from the raw image. The bottom image shows a LLDG reconstructed from an ideal set of wing samples as discussed in Chapter 6.

In the next section, we will present an overview of the problem studied in this dissertation, and a quick survey of different object representation schemes in the literature. An outline of the organization of this thesis concludes the chapter.

1.1 Problem Definition

Prospects for machine understanding of industrial or household scenes have been greatly enhanced in the last ten years by development of better sensors, by successful modeling of certain lower level functions of the various channels involved in vision, and by progress in model-based object recognition. The concept *recognition* implies that there must be some model in memory which is somehow evoked by the sensory data at hand. Models can range in generality from the very specific object definitions used in CAD/CAM [41], through parameterized forms such as the superquadrics [98], to the qualitative models formed by geons and their relations [13]. (Note that a more limited view is being taken here by not considering recognition by classical pattern recognition [33] or recognition by function [125, 113].) The character of the model is not the only crucial issue; the strategy or algorithm for matching the memory model to the sensory data is just as critical [40, 63, 80]. Central to any object recognition system is the issue of object representation and feature extraction; two of the most researched topics in the computer vision-object recognition community.

An approach somewhat opposite to model-based recognition is that of *line drawing analysis*. In this approach, an attempt is made to interpret the 3D structure of surfaces, edges, and vertices by interpretation of the line drawing of object contours in a 2D image [61, 66, 83, 92, 115, 123]. Usually no model database is assumed, only general information about the geometry and topology of objects. As a result, there is a potential for more generality of object definition and for bottom-up processing.

The major we
sumption that the
perfect line drawing
obtainable, ambigu
tion is added (see

The work pres
recognition and so

1. A proposal i
range data vi
(surface) dat
object limbs
crease edges.

2. Construction
object primiti
once derived, s
level system.
representation
this work prov
proven to be co

In the next subse
offer a quick view of

1.1.1 Wing R

Wing representation
tion. Instead of mode

The major weakness in most line drawing analysis schemes derives from the assumption that the line drawings are perfect, when, in fact, years of work show that perfect line drawings cannot be obtained from real data. Moreover, even if they were obtainable, ambiguous interpretations would still arise unless some real 3D information is added (see [84] for one way to do this).

The work presented in this thesis addresses two important problems in object recognition and scene interpretations:

1. A proposal is made for the detection of object primitives in fused intensity-range data via simultaneously fitting intensity contours and the adjacent range (surface) data. The quadric form and its projection is proposed for modeling object limbs and the intersection of quadric forms is proposed for modeling crease edges.
2. Construction of a labeled line drawing graph from the set of locally detected object primitives rather than a perfect line drawing graph. The line drawing, once derived, should provide many features to index into the models of a higher-level system. By organizing data-directed processes to create the line drawing representation, effects of the segmentation problem are greatly reduced. Hence, this work provides a partial solution to the “segmentation problem” which has proven to be combinatorially difficult for most model-based schemes.

In the next subsection, we briefly introduce the *wing representation theory* and offer a quick view of our methodology for extracting *wing features*.

1.1.1 Wing Representation Theory

Wing representation theory was introduced by Chen [26, 27] for 3D object recognition. Instead of modeling objects by parts [53], objects are modeled by set of views

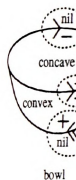


Figure 1.2. Unique
all have the same

composed of both
ple wing is defined
a fragment of ima
mination, symmet
wings are grouped
ilar projected line
bowl, a half grapefr
represented using w

In [28], a comp
representations of ge
conclusion that wing
stability. However, n

A more in-depth
representation offers
that the wing featur
from a raw input sce
segmentation [26]. B
remains to be discov

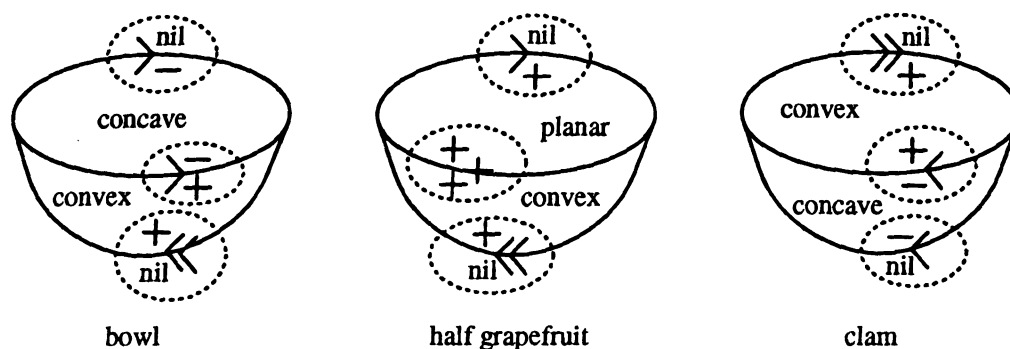


Figure 1.2. Unique wing representation of a bowl, a half grapefruit and a clam, which all have the same line drawing [27].

composed of both *simple wings*, *composite wings* and their spatial structures. A *simple wing* is defined as a triplet that includes a pair of surface patches separated by a fragment of image contour. Through some non-accidental relations such as cotermination, symmetry, parallelism, connectivity, collinearity and curvilinearity, object wings are grouped into *composite wings*. Using wing representation, objects with similar projected line drawings can be distinguished easily. For example, in Figure 1.2 a bowl, a half grapefruit and a clam all have the same line drawing, but can be uniquely represented using wings.

In [28], a computational framework for construction of the multiple-view wing representations of general 3D rigid objects was proposed. With it, was the preliminary conclusion that wing representations have the properties of uniqueness, locality, and stability. However, no satisfactory wing extraction method was proposed.

A more in-depth treatment of wing definition will be given in Chapter 2. Wing representation offers a good basis for an object recognition system. It was conjectured that the wing features are the highest level scene features that can be extracted from a raw input scene without using any specific object models or any prior scene segmentation [26]. But as with other theoretical work, an effective feature extractor remains to be discovered.

In this disserta

1. push forward

2. complete (pa
of detected v

Wings are to b
a local area of int
hypothesized. Veri
problem in which
recovery of surface
parameter estimate
of a wing primitive
adjacent surface pa
from the input imag
surfaces present in

Rather than the
from a input image
terpretation via line
should be produced
raw image, we recor
issue involved is whe
tion. We will explo
and derive reconstru
for complete line dra
scenes will be given.
be provided. To the
tries to simultaneous

In this dissertation, we strive to

1. push forward toward automatic wing detection; and
2. complete (partial) reconstruction of a labeled line drawing graph from the set of detected wings.

Wings are to be extracted from a single registered range and intensity image. In a local area of interest in the fused image, existence of various wing primitives is hypothesized. Verification of a wing hypothesis is posed as a non-linear optimization problem in which the 2D contour constraint, whenever available, is used to aid the recovery of surface parameters. The 2D contour is also useful in generating the initial parameter estimates required by the non-linear optimization technique. Rectification of a wing primitive gives qualitative as well as quantitative information about the adjacent surface patches. No specific object models are used. Wings are extracted from the input image without any pre-processing nor knowledge of the explicit object surfaces present in the scene.

Rather than the usual two-step process (construction of a perfect line drawing from a input image followed by analysis of the derived line drawing) in scene interpretation via line drawing, we reason that a line drawing and its interpretation should be produced concurrently. Specifically, given the set of sampled wings from a raw image, we reconstruct the *labeled line drawing graph* (LLDG) in one step. The issue involved is whether wing representation is adequate for line drawing reconstruction. We will explore the geometric constraints underlying the line drawing graph and derive reconstruction rules based on those constraints. Deterministic algorithms for complete line drawing reconstruction of origami, polygonal and quadric surface scenes will be given. Heuristic algorithms for generating partial line drawings will also be provided. To the best of our knowledge, Trytten [119] is the only other work that tries to simultaneously derive and label the line drawing graph. However, this work

differs from Trytt
classes of objects,

1.2 Repre

Recog

This thesis is about
more popular objec
for the purpose of o
object representati

Generalized C

Generalized cylinde
3D objects. It is d
the *eccentricity* of t
corresponds to the
arbitrary planar sha
axis. The sweeping
eccentricity is the an
a right circular cylin
straight axis with no
90 degree.

The Generalized
rotation, scale and ill
attention in the past.
on the generalized cy

differs from Trytten's in several aspects including our use of fused imagery, different classes of objects, and Trytten's usage of junction catalogue.

1.2 Representational Schemes for 3D Object Recognition

This thesis is about scene representation. In this section, we briefly survey some of the more popular object representation schemes in use by the computer vision community for the purpose of object recognition. [11, 29] provide in-depth surveys of various 3D object representation and recognition techniques.

Generalized Cylinder

Generalized cylinders or generalized cones were introduced by Binford [14] to model 3D objects. It is defined by an *axis*, a *cross-section* function, a *sweeping rule* and the *eccentricity* of the generalized cylinder. The axis is an arbitrary 3D curve and corresponds to the spine of the object being described. The cross-section is of any arbitrary planar shape that may change in shape and size as it is swept along the axis. The sweeping rule governs how the size of the cross-section changes, and the eccentricity is the angle between the axis and the cross-section plane. For example, a right circular cylinder can be represented as a circular cross-section, swept along a straight axis with no change in shape or size of the cross section and with eccentricity 90 degree.

The Generalized cylinder is a volumetric shape descriptor that is invariant to rotation, scale and illumination changes. For this reason, it has received considerable attention in the past. While Marr has proposed an object recognition system based on the generalized cylinder [86], Brooks developed *ACRONYM* [21], a model-based

object recognition

have concentrated

from dense range

Geons

Biederman [13] pro

lons for use in his

by a set of four c

tional/reflective, re

ing/contracting) an

cylinders.

An object is re

structural relations

the relations includ

approximately equa

to G_2 ; (3) Centerin

a geon's surface; (4)

short surface of G_1 a

Biederman's sim

154 million qualitat

geons, respectively a

shows that human la

objects, implying tha

sufficient representat

An edge based p

also proposed. First,

properties such as cur

object recognition system that explicitly models objects as generalized cones. Others have concentrated on extracting generalized cylinders from raw input images ([2, 94] from dense range images and [105] from intensity images).

Geons

Biederman [13] proposed a set of 36 volumetric primitives called *Geons* or *Geometric Ions* for use in his theory of Recognition-by-Components (RBC). Geons are defined by a set of four qualitative features: edge (straight or curved), symmetry (rotational/reflective, reflective, asymmetric), size variation (constant, expanding, expanding/contracting) and axis (straight, curved); and is a sub-class of the generalized cylinders.

An object is represented by a set of geons, denoting its components, and the structural relations among pairs of geons. Given two connected geons G_1 and G_2 , the relations include (1) Relative Size: G_1 could be greater than, smaller than or approximately equal to G_2 ; (2) Verticality: G_1 could be above, below or side connected to G_2 ; (3) Centering: the point of attachment could be centered or off-centered on a geon's surface; (4) Relative size of surfaces at join: the join could be on a long or short surface of G_1 and G_2 .

Biederman's simple calculation showed that approximately 75 thousand and over 154 million qualitatively different objects can be constructed using only 2 and 3 geons, respectively and inter-geon relationships. He argued that liberal estimation shows that human language vocabulary consists of only 30,000 readily discriminable objects, implying that the set of 36 geons along with their inter-geon relations have sufficient representational power.

An edge based procedure for segmenting an image into geon components was also proposed. First, a line drawing of the objects was obtained, from which, edge properties such as curvature, collinearity, symmetry, parallelism and co-termination

were detected. Each component drawing is the cr from range image

Superquadrics

Instead of modeling cylinders, Pentlar thought of as "lun bending, twisting using Boolean operations deformed, reshaped Superquadrics

$$\vec{X}(\eta)$$

where $C_\eta = \cos(\eta)$ meanings. Parameters y and z axes, respectively the z -axis and in the angles of vector \vec{X} in

Recovering superquadrics from the surface patches and ϵ_2) parameters. the literature but all

were detected. The objects were segmented at concave regions and the identity of each component was derived from the edge properties. The extraction of the line drawing is the critical and the most difficult step. Obtaining geon-like parts directly from range images has recently been addressed by Raja [103, 104].

Superquadrics

Instead of modeling object parts by qualitative shapes such as geons or generalized cylinders, Pentland [97] suggested the use of *superquadrics*. The basic parts are thought of as "lumps of clay" which may be deformed and reshaped by stretching, bending, twisting or tapering. These basic "lumps of clay" can then be combined using Boolean operations to form new, more complex prototypes which again can be deformed, reshaped and combined.

Superquadrics are defined by the equation

$$\vec{X}(\eta, \omega) = \begin{pmatrix} a_1 C_\eta^{\epsilon_1} C_\omega^{\epsilon_2} \\ a_2 C_\eta^{\epsilon_1} S_\omega^{\epsilon_2} \\ a_3 S_\eta^{\epsilon_1} \end{pmatrix}, -\frac{\pi}{2} \leq \eta \leq \frac{\pi}{2}, -\pi \leq \omega \leq \pi;$$

where $C_\eta = \cos(\eta)$ and $S_\omega = \sin(\omega)$. The parameters have intuitive geometric meanings. Parameters a_1, a_2 and a_3 affect the size of the superquadrics along the x, y and z axes, respectively. ϵ_1 and ϵ_2 govern the squareness of the superquadrics along the z-axis and in the xy-plane. Parameters η and ω are the latitude and longitude angles of vector \vec{X} in spherical coordinates.

Recovering superquadrics in general position requires estimating 11 parameters from the surface patch. These are location (3), orientation (3), size (3) and shape (ϵ_1 and ϵ_2) parameters. Various schemes for superquadrics fitting have been proposed in the literature but all assume that the images are pre-segmented into regions corre-

sponding to objects
with some interre-

Problems w

A problem comm
of those parts fro
efforts in fitting th
segmentation is re
more, those volun
objects can be sa
2.5D. Blindly mod
cause false alarms

Aspect Graph

In contrast to part
sentation of 3D obj
the aspect graph of
Each node, defined
the object as seen f
views from this com
arc joining two aspe
crossing from one re
all possible stable v
by some aspects and
graph for a tetrahed

sponding to object parts [5, 48, 98, 112]. Superquadric fitting to actual range image with some intermediate segmentation have been demonstrated in [38, 50].

Problems with Existing Parts-Based Representation

A problem common to all the above object representation schemes is that extraction of those parts from a raw image is difficult. Most researchers have concentrated their efforts in fitting those forms to *pre-segmented* images. Parts are rich and sophisticated segmentation is required, whereas wing theory is a theory of segmentation. Furthermore, those volumetric parts model object parts as a 3D solids; hence, no origami objects can be satisfactorily represented. In addition, the images are not 3D but 2.5D. Blindly modeling the backside of an object that is not visible in the image can cause false alarms in model-based recognition systems.

Aspect Graph Representation

In contrast to parts-based representation, aspect graphs are a multiple-view representation of 3D objects. Koenderink and van Doorn [71] introduced the idea of using the aspect graph of topologically distinct views of an object to represent its shape. Each node, defined as an aspect, in the aspect graph represents a “stable view” of the object as seen from some maximal connected region of viewpoint space. Object views from this connected region of viewpoint space appear qualitatively similar. An arc joining two aspects represents a possible transition between two stable views by crossing from one region to another. To complete the definition of an aspect graph, all possible stable views and transitions between such stable views are represented by some aspects and arcs in the aspect graph. Figure 1.3 shows an example aspect graph for a tetrahedron.

Fig

This concept of
scheme for object r
matic generation of
orthographic and pe
theory coupled with
object recognition sy

1.3 Organ

The remainder of thi
define the object dom
line drawing graph (L
will derive the set of
based on those definit
describe the process c

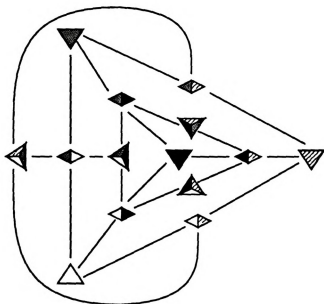


Figure 1.3. Aspect graph of a tetrahedron (from [71]).

This concept of aspect graph, intuitively, seems to be a powerful representation scheme for object recognition. Many research efforts have been devoted to automatic generation of aspect graphs of object models in various domains under both orthographic and perspective projections (for a listing see [19]). Wing representation theory coupled with aspect graph model representation might be a good basis of an object recognition system.

1.3 Organization of the Dissertation

The remainder of this dissertation is organized as follows. In Chapter 2, we formally define the object domain of this research. The formal definitions of object wing, the line drawing graph (LDG), and the labeled line drawing graph (LLDG) are given. We will derive the set of all possible wing models and their mathematical representation based on those definitions. We will also explain the setup of our imaging system and describe the process of range and intensity images fusion.

Detection of
from objects pro
propose simultan
linear equations
on synthetic data
surface data alon
needed to achieve
set of real images
limb or crease edge
consistent with th
We conclude that
wings from fused i
The collection o
tion algorithms dep
algorithms, POLY-
ing graph of polygo
is that the range is
geometric constrain
Junction catalogues
tion algorithms. Th
several of which hav
In Chapter 5, we
idealistic assumption
occur in the input
inaccuracies are tol
construction of the
experimentally show

Detection of wings is treated in Chapter 3. It is well-known that edge information from objects provides strong clues about the local shape of surface region [70]. We propose simultaneous boundary and surface fitting with few sample points. The non-linear equations for fused fitting are derived. Results from Monte Carlo experiments on synthetic data show that fused fitting is superior to fitting either boundary or surface data alone. Moreover, only a relatively small number of data points are needed to achieve quality parameter estimates. Results of wing detection on a large set of real images are analyzed. We observe that fitting fused data near an object limb or crease edge is advantageous and more accurate than fitting surface data alone, consistent with the finding from the Monte Carlo experiments on synthetic images. We conclude that our wing detector, though not perfect, can reliably extract object wings from fused imagery without any special pre-processing.

The collection of wing samples form the input to the line drawing graph reconstruction algorithms depicted in Chapters 4 - 6. In Chapter 4, we present two deterministic algorithms, POLY-1 and POLY-2, that can reconstruct the *perfect* labeled line drawing graph of polygonal scenes under some idealistic assumptions. In POLY-1, the key is that the range image must be available for re-sampling whereas in POLY-2, the geometric constraints of LDG are deduced and utilized to guide the reconstruction. Junction catalogues and fallible heuristic rules are not used in these two reconstruction algorithms. The algorithms are tested on over 15 scenes, both synthetic and real; several of which have appeared as problematic scenes in the literature.

In Chapter 5, we present a heuristic algorithm, POLY-3, that operates without the idealistic assumptions as in Chapter 4. Instead, missing and spurious wing samples occur in the input wing set. Errors introduced by computation and measurement inaccuracies are tolerated. A set of heuristic rules are derived to perform the reconstruction of the LLDG given that the input wing samples are incomplete. We experimentally show that algorithm is capable of reconstructing the complete LLDG

if the input wing

defective. Results

In Chapter 6,

curved objects. W

tions, POLY-1 and

There are two ma

main: wing cluste

introduced that by

solving the wing e

LLDGs from synth

the dissertation by

for future research.

if the input wing sample is complete and a partial LLDG if the input wing set is defective. Results from both synthetic and real images are provided.

In Chapter 6, the aforementioned algorithms are extended to handle scenes with curved objects. We argue and show by example that under the same idealistic assumptions, POLY-1 and POLY-2 can be directly applied with only minor modifications. There are two major difficulties in direct extension of POLY-3 for the expanded domain: wing clustering and wing extension. A new heuristic algorithm, QUAD-2 is introduced that bypasses the step of wing clustering. Future research directions for solving the wing extension problem are also suggested. Examples of reconstructed LLDGs from synthetically generated wing samples are shown. Chapter 7 concludes the dissertation by summarizing the contributions of this research and pointing ways for future research.

CHAP

Formal

and W

2.1 Intro

There has been much work in the area of surface fitting and line drawing reconstruction. A 3D surface model (3D surface

In the following thesis are defined. The mathematical

Formal definitions

stated. We will present a drawing graph and

analysis. In addition, images for use as test

CHAPTER 2

Formalization of Object Domain and Wing Models

2.1 Introduction

There has been much work in the analysis of line drawings and a lot of work in the area of surface fitting. In this chapter we will lay the foundation for the wing detection and line drawing reconstruction techniques, which are based largely on fitting a wing model (3D surface model and 2D contour model) to fused range and intensity data.

In the following sections, terms that are essential to the understanding of this thesis are defined. In particular, the object domain and wing models are characterized. The mathematical representation of the wing models and their derivations are given. Formal definitions of a *line drawing graph* and *labeled line drawing graph* are also stated. We will point out the differences between our definition of a *labeled line drawing graph* and the line labeling commonly found in the work of line drawing analysis. In addition, the process of acquiring registered fused range and intensity images for use as test data is also described.

2.2 Obj

In this research,

a given viewpoi

Appendix A. W

and intensity ima

sensed, regardless

in terms of the su

would produce “u

The very first

research is that i

polyhedron or pol

restricted. Their

definition excludes

[55] complex defin

cannot be sensed.

related by Lakatos

be both mathemat

Before defining

catalogue nor the o

objects considered

δ -quadric surfaces.

[66] are allowed as

which have satisfact

2.2 Object Domain Definitions

In this research, the input is a registered range and intensity image of the scene from a given viewpoint. The sensing equipment and the set-up is described in detail in Appendix A. With our set-up, which has a single camera for capturing both range and intensity images, only those surfaces in the scene visible to the camera are being sensed, regardless of the shape of the surfaces. Therefore, we define the object domain in terms of the surfaces that bound the object and the legal viewpoints as those that would produce “usable” registered range and intensity image.

The very first problem encountered and an important lesson learned from this research is that it is not easy to get the appropriate general definition for even a polyhedron or polygon! The definition used by Huffman and Clowes [30, 61] is too restricted. Their block world includes objects with only trihedral vertices. That definition excludes many common polyhedral and all origami objects. Hoffmann’s [55] complex definitions support CAD operations well but involve structures which cannot be sensed. After reading the fascinating history of “Eulerian polyhedra” related by Lakatos[74] we are less ashamed of our struggle for definitions that would be both mathematically and practically effective.

Before defining our objects, it is important to note that we do not need a junction catalogue nor the object domain restrictions that such a catalogue would imply. The objects considered in this thesis are those whose faces are either δ -polygons and/or δ -quadric surfaces. The objects do not have to be solid. Objects from origami world [66] are allowed as long as the faces are all δ -polygons. First, we define polygons which have satisfactory *size properties* if properly viewed by the sensor.

2.2.1 δ -po

Definition 2.1

bounded by straig

that there exists a

nonzero area of b

of other polygon a

The radius, r ,

imposes a minimum

P to all other edge

short for accurate

polygon is closed,

condition is actual

case in Theorem 4.

Note that more

main difference bet

any closed polygon

exists a point on e

edges is greater tha

so are bowties. No

segments. Some exa

2.2.2 δ -quad.

Extension of the no

polygonal case, the

dealing with a curve

what the vertices are

2.2.1 δ -polygons

Definition 2.1 *A δ -polygon is a closed region of finite extent in a 3D plane bounded by straight edge segments; moreover, (a) each edge contains a point P such that there exists an open neighborhood N_r of radius $r > \delta > 0$ about P which includes nonzero area of both the interior and exterior of the polygon, and includes no points of other polygon edges; (b) every vertex has an even number of incident edges.*

The radius, r , of the open neighborhood N_r in (a), of minimum length δ , implicitly imposes a minimum length on each edge of the polygon and a minimum distance from P to all other edges. Basically, it dismisses all objects that may be too thin or too short for accurate range sensor registration and wing detection. To ensure that the polygon is closed, condition (b) requires an even number of incident edges. This condition is actually implied by the other restrictions. We will show that this is the case in Theorem 4.2. It is explicitly stated here for emphasis.

Note that more than 2 coincident edges are allowed. Intuitively speaking, the main difference between a δ -polygon and the “usual” polygon is that a δ -polygon is any closed polygon such that (1) the length of each edge is at least δ , and (2) there exists a point on every edge such that the distance between that point to all other edges is greater than δ . With the above definition, not only are holes allowed, but so are bowties. Non- δ -polygons include those polygons with a cut or dangling line segments. Some examples of δ -polygons and non- δ -polygons are shown in Figure 2.1.

2.2.2 δ -quadric surfaces

Extension of the notion of δ -polygon to δ -quadric is not straightforward. In the polygonal case, the definition of an edge is intuitively clear. This is not so when dealing with a curved surface patch. Before defining an *edge*, we must first define what the vertices are. The vertices of a quadric surface patch are those points on

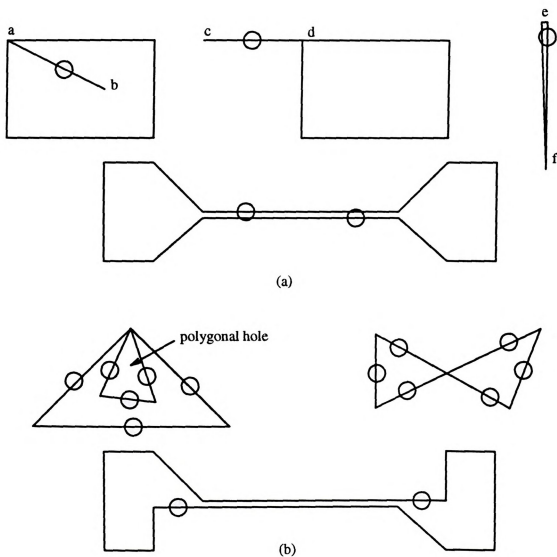


Figure 2.1. Examples of valid and invalid δ -polygons. (a) non- δ -polygons: \overline{ab} and \overline{cd} have points having neighborhood with zero exterior/interior polygonal area; \overline{ef} has no point P that would satisfy the open neighborhood property. (b) δ -polygons. Open neighborhood about a point P is represented by \bigcirc .

the bounding cu
where the curva
segments separa
restriction on th
that the minimu
curve is greater t

Definition 2.2

such that there e
quadric surface p
lines, $L = \{l_1, l_2\}$
such that an open
area of both the i
 $j \neq i$, and (3) on
incident with an e

In this thesis,
surfaces. Specifica
surfaces will be all
projection of those
linear, circular or
boundaries will alw
regardless of the vie

Using this defini
than $\delta/2$. Otherwi
projected bounding
Likewise, a right ci
greater than δ . Thu

the bounding curve where the curvature or tangent is discontinuous and those points where the curvature changes sign (points of inflection). The *edges* are the boundary segments separated by the vertices. Since a closed curve is now possible, an additional restriction on the coverage of the open neighborhood N_r is also needed to guarantee that the minimum of the maximum distance between any two points on the closed curve is greater than δ .

Definition 2.2 *A δ -quadric is a closed 3D quadric surface patch of finite extent such that there exists a viewing direction in 3D such that the 2D projection of the quadric surface patch along it is a planar image region R bounded by a set of (curved) lines, $L = \{l_1, l_2, \dots\}$; moreover, for each $l_i \in L$, (a) there exists a point P on l_i , such that an open neighborhood N_r of radius $r > \delta > 0$ about P includes (1) nonzero area of both the interior and exterior of the planar region R , (2) no points of l_j if $j \neq i$, and (3) only a proper subset of points from l_i ; and (b) every junction of R is incident with an even number of l_i 's.*

In this thesis, we will restrict δ -quadric surfaces to a special subclass of quadric surfaces. Specifically, only planar, spherical, circular cylindrical and circular conical surfaces will be allowed and each edge must be of type linear, circular or elliptic. The projection of those edges onto the 2D image plane results in (curved) lines of type linear, circular or elliptic. Furthermore, given those four surface types, their limb boundaries will always be projected to straight or circular lines in the image plane regardless of the viewpoint.

Using this definition, a spherical patch would be a δ -quadric if its radius is greater than $\delta/2$. Otherwise, any neighborhood N_r of radius $r > \delta$ on any point on the projected bounding curve would include the entire boundary (see Figure 2.2 (b)). Likewise, a right circular cylinder must have radius greater than $\delta/2$ and height greater than δ . Thus, the two sides of a penny are δ -quadrics if $r > \delta/2$, but not

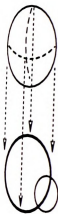


Figure 2.2. Exa
quadrics. Open m

the cylindrical su
cylindrical surface

valid view becaus

area (*i.e.*, no visib

Figure 2.2 (b).

For simplicity,

ably with δ -polygo

2.2.3 Sample

We are now ready

There are 2 types of

1. Origami or po

2. Curved objects

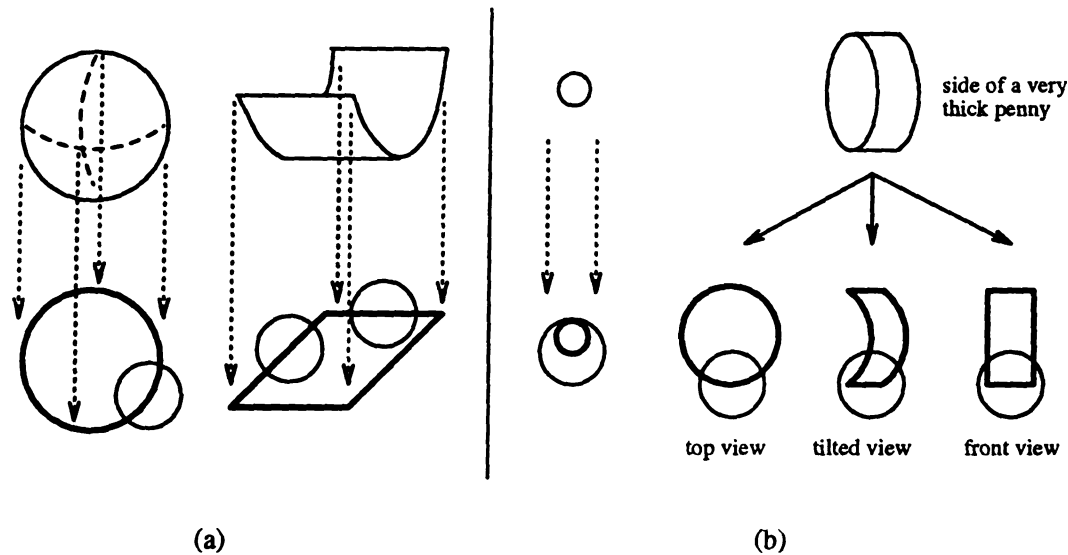


Figure 2.2. Examples of valid and invalid δ -quadrics. (a) δ -quadrics. (b) Non- δ -quadrics. Open neighborhood about a point P is represented by \bigcirc .

the cylindrical surface because its height is too short. Note the top view of the cylindrical surface projects to a circle which has radius $r > \delta/2$ but this is not a valid view because the projection of the cylindrical surface yields a region with 0 area (*i.e.*, no visible region). Possible views of this cylindrical surface are depicted in Figure 2.2 (b).

For simplicity, the terms polygons and quadric surfaces will be used interchangeably with δ -polygons and δ -quadric surfaces.

2.2.3 Sample of Objects in the Object Domain

We are now ready to define the type of objects that belong to the object domain.

There are 2 types of objects in the object domain:

1. Origami or polyhedral objects where every face is a δ -polygon.
2. Curved objects whose bounding faces are δ -quadrics.



Figure 2.3. (a) E
not in the object

The first class
objects may be bo
struction of scenes
the curved surface
how a complete lin
polygons. Example

class are shown in I

Note that the p
surface is too short t
may project to the s
is shorter than δ . W
an accidental view is

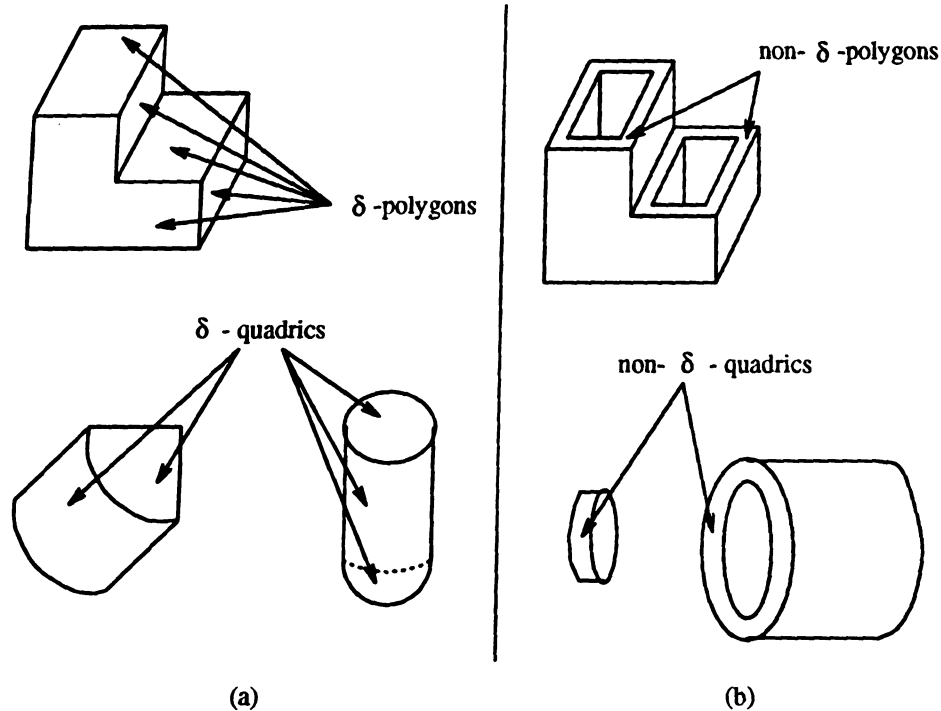


Figure 2.3. (a) Examples of objects in the object domain. (b) Examples of objects not in the object domain.

The first class of objects contains only planar surfaces, whereas the second class of objects may be bounded by curved surfaces. Wing detection and line drawing reconstruction of scenes involving objects in the first class are easier than those involving the curved surface objects. In fact, in Chapter 4 we will give an algorithm showing how a complete line drawing may be derived from a set of ideal wings sampled from polygons. Examples of objects in each class as well as objects that belong to neither class are shown in Figure 2.3.

Note that the penny is not in the object domain. The height of the cylindrical surface is too short to qualify it as a δ -quadric. The test tube, although a valid object, may project to the same line drawing where the height of the test tube in the image is shorter than δ . We shall call this view an *accidental view*. The formal definition of an accidental view is given in the next section.

2.3 Acc

Wing detection
task. It operate
pre-determined
edge and adjace
the same surfac
varying sizes. F
the plane projec
lost. Detection o
yields very short
those situations a
next two definitio
accidental views o

Definition 2.3 [
(quadric) scene is

1. there exists a
quadric) from
2. every non-jun
labeled with o

Note that from
image plane has len

Definition 2.4 An
Sensing Property is

This definition of
dental view is define

2.3 Accidental and Non-Accidental Viewpoints

Wing detection, as described in the next chapter, is a low-level computer vision task. It operates on fused range and intensity data of the scene as seen from some pre-determined viewpoint. Success of wing detection depends on how much of the edge and adjacent surface are visible in the image plane. With different viewpoints, the same surface patch/edge may project onto the image plane as regions/lines of varying sizes. For example, a planar surface seen from a viewpoint co-planar with the plane projects onto the image plane as a straight line; the surface information is lost. Detection of wings is impossible in this and other cases where the projection yields very short line segments and/or small regions in the image plane. We define those situations as consequences of sensing from *accidental* viewing directions. The next two definitions define the *sensing property* of viewpoints and *accidental/non-accidental views* of the scene.

Definition 2.3 [Sensing Property] *A non-accidental view of a polygonal (quadric) scene is a view with all of the following properties :*

1. *there exists some $\epsilon > 0$ such that the visible projection of every δ -polygon (δ -quadric) from 3D is an ϵ -polygon (ϵ -quadric) embedded in the image plane;*
2. *every non-junction point of any line segment in the image plane can be correctly labeled with one and only one wing label.*

Note that from Definitions 2.1 and 2.2 it follows that each line segment in the image plane has length at least ϵ .

Definition 2.4 *An accidental view is a view where one of the conditions of the Sensing Property is violated.*

This definition of accidental view differs from that of [83, 91], in which an accidental view is defined as the orthographic projection of 3D edges under an unstable



Figure 2.4. (a) π - π correlation under [83, 91] accidental view i

viewpoint. A view of a line drawing changes as the viewpoint changes. In this case, the face can be (partially) visible edge resulting in a representation of the drawing graph will

As an example, the accidental view of a line segment to the line segment

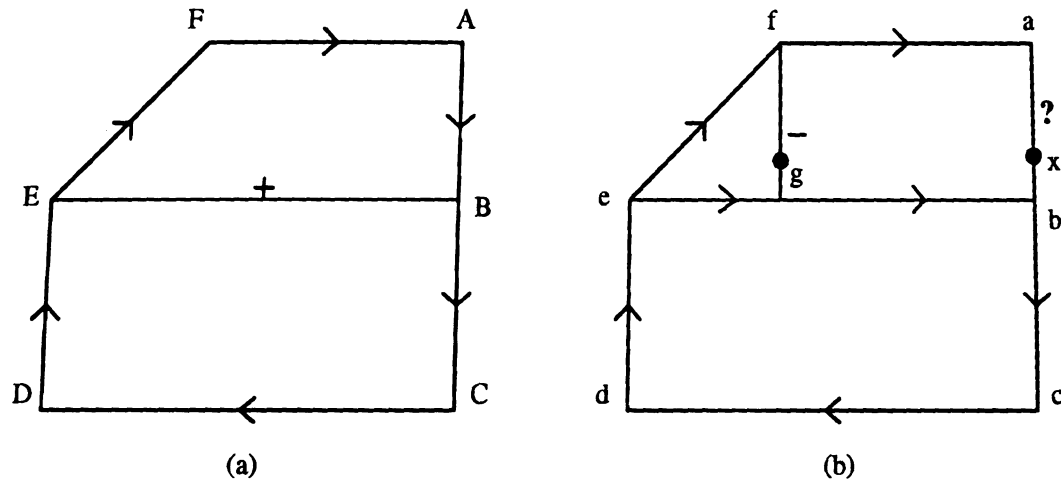


Figure 2.4. (a) A non-accidental view of a solid box, which is considered accidental under [83, 91] (b) An accidental view of a topless box. (This would not be an accidental view if the side containing a , b , c were removed from the box!)

viewpoint. A viewpoint is defined to be unstable if the structure of the projected line drawing changes as the viewpoint is moved within some open set of the Gaussian sphere. In this thesis, a non-accidental view is one where each visible edge and surface can be (partially) sensed. Without this condition, an edge detector might miss a visible edge resulting in no detected wings for that edge. As a result, the corrupted wing representation of the scene will be incomplete and the reconstruction of the *line drawing graph* will be much more difficult.

As an example, Figure 2.4 (a) shows a view of a solid box that would be considered accidental under [83, 91] but is perfectly legal by Definition 2.3. Figure 2.4 (b) is an accidental view of a topless box. It is accidental because the 3D edge corresponding to the line segment marked "?" and its adjoining surface(s) cannot be sensed.

2.4 The

In this research
it is not import
are either δ -pol
3D polygons/qu

Definition 2.5

forms a planar g

In this regard
including [30, 61
naming conventio
LDG has *regions*,
depending on the

In general, line
LDG to be an ima
boundary (\gg) or c
in the LDG, too.

Definition 2.6 A

a labeled line dra

The complexity
is purposely left uns
to the application ne
interpretation ($\{+, -$
circular, parabolic)
pose information of
used in this research

2.4 The Line Drawing Graph (LDG)

In this research, objects are assumed to be composed of δ -polygonal/ δ -quadric faces; it is not important whether they are regarded as 3D solids or just 2D surfaces. Faces are either δ -polygons or δ -quadrics. A *view* of a polygonal/quadric scene is a set of 3D polygons/quadrics with some viewing direction.

Definition 2.5 *The projection of edges of objects in a scene with hidden line removal forms a planar graph called the line drawing graph (LDG).*

In this regard, the LDG is the starting point of much line labeling analysis research including [30, 61, 66, 83, 115, 123]. Following previous authors, we will adopt the naming convention that faces in the 3D world have *edges*, and *vertices*, while the LDG has *regions*, *line segments*, and *junctions*. Note that line segments in the LDG depending on the object domain are not necessarily straight.

In general, line drawing analysis attempts to interpret every line segment in the LDG to be an image of a convex crease (+), concave crease (-), self-occluding surface boundary (\gg) or occluding surface boundary ($>$). Here, we strive to label the regions in the LDG, too.

Definition 2.6 *A LDG where each line segment and each region is labeled, is called a labeled line drawing graph (LLDG).*

The complexity of line and region labels are not specified in the above definition. It is purposely left unspecified so the complexity of the labels can be adjusted according to the application need. For our purposes, the line label not only must contain the 3D interpretation ($\{+, -, \gg, >\}$) of the segment it must also capture the shape ($\{linear, circular, parabolic\}$) of the segment. The region labels will have the shape and the pose information of their 3D surface counterparts. The exact line and region labels used in this research is the topic of next section.

2.5 Wh

The notion of w
dron representa
primitives for 3
to support anal
similar to that

Definition 2.7
*that can appear
labels for the lin
appear in the im
from the sample*

Definition 2.8
*called the wing
that project onto
thermore, a wing*

Since each win
will often refer to
Note that this defin
and line labels. If
would consists of p
in most line labelin
line segment in the
other works on line
In Chen's work [K
concave, saddle), a

2.5 What are Wings?

The notion of wing was first conceived by Baumgart [9] as a data structure for polyhedron representation for computer vision. Chen [27] later defined object wings as $2\frac{1}{2}D$ primitives for 3D object recognition and showed that wing representation is suitable to support analysis of scenes of general objects. Here, the notion of wings will be similar to that of Chen.

Definition 2.7 *Let $S = \{s_1, s_2, \dots, s_m\}$ denote the set of different labels of surfaces that can appear in the scene and let $C = \{c_1, c_2, \dots, c_n\}$ denote the set of different labels for the lines (2D projection of the 3D edge), both straight and curved, that can appear in the image plane. Then a wing primitive, is defined as a triplet drawn from the sample space formed by the set cross product $S \times C \times S$.*

Definition 2.8 *A wing in an image is defined as a fragment of image contour, called the wing contour along with the surface patches, called the wing surfaces, that project onto the regions adjoining the wing contour fragment in the image. Furthermore, a wing can be uniquely labeled as one of the wing primitives.*

Since each wing in the image is uniquely labeled by one of the wing primitives, we will often refer to this unique wing primitive when we speak of wings in the image. Note that this definition of wing places no restriction on the complexity of the surface and line labels. If $S = \{NIL\}$ then the image is represented by a set of wings which would consist of purely line labels. If the line labels are restricted to labels found in most line labeling research (e.g., Malik [83]) and if each wing contour is a whole line segment in the image, then the wing representation is equivalent to that used in other works on line labeling (Figure 2.5 (a)).

In Chen's work [27], there are 5 surface types, {null (background), planar, convex, concave, saddle}, and 4 contour types, {silhouette, jump, convex crease, concave



(a)

Figure 2.5. Rep
catalogue (b) C

crease). Theore
definition. Howe
physically realiz
image using this
the wing contour
3 different wing
LLDG wing repr
capture explicit v
and surface pose

2.5.1 Wing

In defining the ob
restricted to {plan
labeling schemes (a
as surface labels. T

$$S = \{ \text{PLN}(a,b,d) \}$$

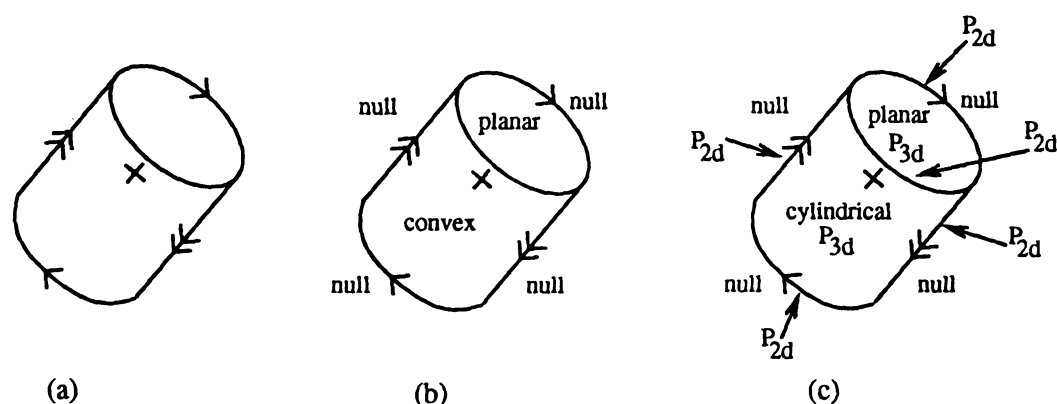


Figure 2.5. Representation of a coke can. (a) Line labeling using Malik's [83] junction catalogue (b) Chen's wing [27] representation. (c) LLDG representation.

crease}. Theoretically there are $5 \times 4 \times 5 = 100$ different wing primitives in his definition. However, as Chen pointed out, only 34 of those 100 wing primitives are physically realizable: that is there are only 34 different types that can appear in the image using this definition of surface and contour labels. For example, if we make the wing contour to take full length, then a coke can would have 5 wings but only 3 different wing primitives are needed to label them all (see Figure 2.5 (b)). The LLDG wing representation (Figure 2.5 (c)) differs from Chen in that we want to capture explicit wing contour/surface shape information as well as the contour label and surface pose information.

2.5.1 Wing Primitives

In defining the object domain, the set of bounding surfaces of all objects have been restricted to {planar, spherical, cylindrical, conical}. Instead of using other surface labeling schemes (as in [27]), we use only those 4 surface types and their pose equations as surface labels. Thus,

$$\mathbf{S} = \{ \text{PLN}(a,b,d), \text{SPH}(x_0,y_0,r_0), \text{CYL}(x_0,y_0,r_0,\alpha,\beta), \text{CON}(x_0,y_0,z_0,r_0,\alpha,\beta), \text{NIL} \}$$

where NIL indicates the background surface or don't care. The equation of each of the 4 surfaces in general position will be given in a later section.

The wing contour label is to have information about both the shape of the curve as well as the type of the curve. Instead of defining wing contour to have shape of general quadric polynomial, it is represented by 3 types of explicit 2D curves, namely linear, circular and parabolic. Since the limb boundaries of cylinders or cones and spheres project to straight lines and circles, respectively, linear and circular curves are used as wing contour models. Note that some non-limb edges may also project to a straight or circular curve. Projection of all other curved edges are *locally* modeled by parabolic curves in 2D. Besides labeling each wing contour with explicit shape information, 3D interpretation of the line segment is also recorded. Following the symbols used in the line drawing analysis literature [83], each wing contour is also labeled as a jump ($<$), convex crease ($+$), concave crease ($-$) or limb (\ll). Thus 12 different labels are possible for wing contours.

Half-wings

As stated earlier, part of the goal of this research is to derive a reliable wing detector. Instead of defining wings as three-tuples, (S_i, C_k, S_j) where $S_i, S_j \in \mathbf{S}$ and $C_k \in \mathbf{C}$, they can be expressed as the union of two-tuples, $(S_i, C_k) \cup (S_j, C_k)$, with a common wing contour. We shall call each of those two-tuples as *half-wings*. The problem of wing detection in an image are sometimes reduced to the problem of detecting two individual half-wings followed by a merge operation. This is true if the edge contour that separates the two surfaces is either jump or limb (depth discontinuity).

With the previously defined wing surface and contour labels, a complete list of half-wing primitives are generated and given in Table 2.1. Note that for each half-wing, the wing contour acts as a separator between the two wing surfaces. The wing contour lies on exactly one of the two wing surfaces. Thus, a half wing could have

a

sp

win

are

2.3

Win

of th

wing

proje

vital

recove

edge p

wing c

Table 2.1. List of all 60 half-wing primitives

Planar:	(PLN+LN _{<<})	(PLN+LN _{<})	(PLN+LN ₊)	(PLN+LN ₋)
	(PLN+CC _{<<})	(PLN+CC _{<})	(PLN+CC ₊)	(PLN+CC ₋)
	(PLN+PB _{<<})	(PLN+PB _{<})	(PLN+PB ₊)	(PLN+PB ₋)
Spherical:	(SPH+LN _{<<})	(SPH+LN _{<})	(SPH+LN ₊)	(SPH+LN ₋)
	(SPH+CC _{<<})	(SPH+CC _{<})	(SPH+CC ₊)	(SPH+CC ₋)
	(SPH+PB _{<<})	(SPH+PB _{<})	(SPH+PB ₊)	(SPH+PB ₋)
Cylindrical:	(CYL+LN _{<<})	(CYL+LN _{<})	(CYL+LN ₊)	(CYL+LN ₋)
	(CYL+CC _{<<})	(CYL+CC _{<})	(CYL+CC ₊)	(CYL+CC ₋)
	(CYL+PB _{<<})	(CYL+PB _{<})	(CYL+PB ₊)	(CYL+PB ₋)
Conical:	(CON+LN _{<<})	(CON+LN _{<})	(CON+LN ₊)	(CON+LN ₋)
	(CON+CC _{<<})	(CON+CC _{<})	(CON+CC ₊)	(CON+CC ₋)
	(CON+PB _{<<})	(CON+PB _{<})	(CON+PB ₊)	(CON+PB ₋)
Background:	(NIL+LN _{<<})	(NIL+LN _{<})	(NIL+LN ₊)	(NIL+LN ₋)
	(NIL+CC _{<<})	(NIL+CC _{<})	(NIL+CC ₊)	(NIL+CC ₋)
	(NIL+PB _{<<})	(NIL+PB _{<})	(NIL+PB ₊)	(NIL+PB ₋)

a planar surface while the wing contour is a projection of a limb from a cylinder or sphere ((PLN+LN_{<<}) or (PLN+CC_{<<})). For completeness, Table 2.2 lists all those wings that can be detected as a whole rather than as two half-wings. Basically, these are wings that involve crease edges.

2.5.2 Derivation of Wing Contour Equations

Wing detection as to be described in the next Chapter, is to proceed by fitting each of the wing models to the sampled surface/contour points. In the case of jump wings (*i.e.*, wing contour corresponds to a jump edge), sample points along the edge projection do not facilitate the wing recovery process. However, those 2D points are vital for the recovery of wings involving limb boundaries and crease edges. To aid the recovery of wings by fitting the wing contour model to the sample points from the edge projection, we must first establish the connection between the equations of the wing contour and the wing surface(s).

su

th

wi

and

pro

gray

In s

2.5

The

P_3

Table 2.2. List of 38 full wing primitives

(PLN+LN ₊ +PLN)		
(PLN+LN ₋ +PLN)		
(PLN+LN ₊ +CYL)	(PLN+CC ₊ +CYL)	(PLN+PB ₊ +CYL)
(PLN+LN ₋ +CYL)	(PLN+CC ₋ +CYL)	(PLN+PB ₋ +CYL)
(PLN+LN ₊ +SPH)	(PLN+CC ₊ +SPH)	(PLN+PB ₊ +SPH)
(PLN+LN ₋ +SPH)	(PLN+CC ₋ +SPH)	(PLN+PB ₋ +SPH)
(PLN+LN ₊ +CON)	(PLN+CC ₊ +CON)	(PLN+PB ₊ +CON)
(PLN+LN ₋ +CON)	(PLN+CC ₋ +CON)	(PLN+PB ₋ +CON)
(CYL+LN ₊ +SPH)	(CYL+CC ₊ +SPH)	(CYL+PB ₊ +SPH)
(CYL+LN ₋ +SPH)	(CYL+CC ₋ +SPH)	(CYL+PB ₋ +SPH)
(CYL+LN ₊ +CON)	(CYL+CC ₊ +CON)	(CYL+PB ₊ +CON)
(CYL+LN ₋ +CON)	(CYL+CC ₋ +CON)	(CYL+PB ₋ +CON)
(SPH+LN ₊ +CON)	(SPH+CC ₊ +CON)	(SPH+PB ₊ +CON)
(SPH+LN ₋ +CON)	(SPH+CC ₋ +CON)	(SPH+PB ₋ +CON)

In the following sections, the equation of planar, spherical, cylindrical and conical surfaces in general position are given. These equations will be denoted as P_{3d} . From these equations, the projections of the limb boundaries of sphere, cylinder and cone, which are circular, linear and linear, respectively, onto the image plane are derived and are denoted as P_{2d} . Finally, we show the equations of the wing contours that are projections of crease edges. But first, we show how to derive the equation of the orthographic projection of the limb edge from the equation of the general quadric surface. In so doing, we will force the two equations to have the same set of parameters.

2.5.3 Deriving P_{2d} from P_{3d}

The general equation of a quadric surface takes the form

$$P_{3d}(x, y, z) = ax^2 + by^2 + cz^2 + dxy + exz + fyz + gx + hy + iz + j = 0. \quad (2.1)$$

T

wh

Eq

of t

from

that

F

term

the d

in pra

data.

between

To find the equation of the projection of the limbs, the points belonging to the limb boundary must be identified. Note that a point on the quadric surface is on the limb if and only if the surface normal at that point is orthogonal to the projection direction. The surface normal at a given point (x, y, z) is simply

$$\mathbf{N}_{P_{3d}}(x, y, z) = \left(\frac{\partial P_{3d}}{\partial x}, \frac{\partial P_{3d}}{\partial y}, \frac{\partial P_{3d}}{\partial z} \right).$$

Assuming that the viewing direction is along the z -axis and that the viewpoint is at $(0, 0, z_v)$, the projection direction for a given point (x, y, z) is then

$$\mathbf{P}_{proj}(x, y, z) = (x, y, z - z_v).$$

Thus, the limbs must lie on the surface defined by the equation

$$\mathbf{P}_{proj} \cdot \mathbf{N}_{P_{3d}} = x(e z_v + g) + y(f z_v + h) + z(2c z_v + i) + (i z_v + 2j) = 0 \quad (2.2)$$

which is planar.

Finally, by solving Equation 2.2 for z in terms of x and y and substituting back into Equation 2.1 and taking the limit as $z_v \rightarrow \infty$, we arrive at P_{2d} , which is the equation of the limb, *orthographically* projected onto the image plane. The P_{2d} equation derived from a general P_{3d} surface equation is long and is omitted here. It should be noted that only simple algebraic manipulations are needed in deriving the final form.

Fitting the general P_{3d} and P_{2d} equations to fused data, which have many complex terms, is computationally expensive [73]. Furthermore, the error model for computing the distance between the fitted model and the data points can only be approximated in practice [24, 73]. Our approach is to fit explicit categorical surface models to fused data. With explicit surface models, we are able to derive and use the exact distance between data points and the model in evaluating the goodness of fit. Moreover, there

are fewer parameters to fit. Below, we give the P_{3d} and the derived P_{2d} limb projection equations of spherical, cylindrical and conical models. This is followed by equations of the projections of crease edges formed by two planar, a planar and a spherical, and a planar and a cylindrical surface.

Spherical Surfaces

A sphere in general position has the equation

$$P_{3d}^{sph}(x, y, z; x_0, y_0, z_0, r_0) = (x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 - r_0^2 = 0 \quad (2.3)$$

There are four parameters: three translational (x_0, y_0, z_0) and radius r_0 . It is obvious that the orthographic projection of the contour is a circle. By applying the procedure outlined above, we arrive at the equation of this circle.

$$P_{2d}^{sph}(x, y, z; x_0, y_0, r_0) = (x - x_0)^2 + (y - y_0)^2 - r_0^2 = 0 \quad (2.4)$$

Note that only three of the four P_{3d}^{sph} parameters are present in P_{2d}^{sph} . This means that fitting P_{2d}^{sph} to contour points leaves one free parameter (*i.e.*, z).

Cylindrical Surfaces

The circular cylindrical model has 5 parameters: two translational (x_0, y_0), two rotational (α, β) and the radius (r_0). We derived P_{3d} for the cylindrical model as follows. Take a cylinder with radius r_0 and the axis coincident with the z -axis. Translate the cylinder along the x -axis by x_0 and along the y -axis by y_0 . Using $(x_0, y_0, 0)$ as the origin of the new coordinate frame, rotate about the x -axis by angle α and then rotate about the new y -axis by angle β . The implicit quadric equation of the cylinder in the new position can then be expressed in the original coordinate frame as:

$$P_{3d}^{cyl}(x, y, z; x_0, y_0, r_0, \alpha, \beta) = ((y - y_0) \cos(\alpha) + z \sin(\alpha))^2 + \\ ((x - x_0) \cos(\beta) + (y - y_0) \sin(\alpha) \sin(\beta) - z \cos(\alpha) \sin(\beta))^2 - r_0^2 \quad (2.5)$$

The orthographic projection of the limbs are two straight lines. The P_{2d} derived using the method outlined above is a quadric equation representing the two parallel lines. By solving for one of the variables, we have the equation of each of the lines.

$$P_{2d}^{cyl}(x, y; x_0, y_0, r_0, \alpha, \beta) = (x - x_0) (\cot(\beta) \sin(\alpha)) + (y - y_0) \pm \\ \frac{\sqrt{r_0^2 (6 - 2 \cos(2\alpha) - \cos(2(\alpha - \beta)) - 2 \cos(2\beta) - \cos(2(\alpha + \beta)))^3 \csc(\beta)^2}}{2^{\frac{3}{2}} (\sin(\alpha)^2 + \cos(\alpha)^2 \sin(\beta)^2)} \quad (2.6)$$

Conical Surfaces

The circular cone model has 6 parameters: three translational (x_0, y_0, z_0) , two rotational angles (α, β) and the height of the cone at unit radius (d_0) . To find the equation of the cone in general position, we first assume that the circular cone is in the upright position (*i.e.*, has its vertex at the origin and the axis coincident with the positive z-axis). By translating the cone along the x-, y- and z-axis by x_0, y_0, z_0 , respectively, then rotating about the new x- and y-axis by α, β as in cylindrical case, we arrive at the equation of the cone in general position.

$$P_{3d}^{con}(x, y, z; x_0, y_0, z_0, d_0, \alpha, \beta) = U^2 + V^2 - W^2/d_0^2, \quad (2.7)$$

$$\begin{aligned} U &= (x - x_0) \cos(\beta) + (y - y_0) \sin(\alpha) \sin(\beta) - (z - z_0) \cos(\alpha) \sin(\beta) \\ \text{where } V &= (y - y_0) \cos(\alpha) + (z - z_0) \sin(\alpha) \\ W &= (x - x_0) \sin(\beta) + (y - y_0) \sin(\alpha) \cos(\beta) + (z - z_0) \cos(\alpha) \cos(\beta) \end{aligned}$$



After expanding and rewriting P_{3d}^{con} as

$$P_{3d}^{con}(x, y, z; x_0, y_0, z_0, d_0, \alpha, \beta) = ax^2 + by^2 + cz^2 + dxy + exz + fyz + gx + hy + iz + j, \quad (2.8)$$

derivation of the equation of the projection of the limb edges is straightforward.

$$P_{2d}^{con}(x, y, z; x_0, y_0, z_0, d_0, \alpha, \beta) = kx^2 + ly^2 + mxy + nx + oy + p, \quad (2.9)$$

where

$$\begin{aligned} k &= a - \frac{e^2}{4c}, & l &= b - \frac{f^2}{4c}, & m &= d - \frac{ef}{2c}, \\ n &= g - \frac{ei}{2c}, & o &= h - \frac{fi}{2c}, & p &= j - \frac{i^2}{4c}, \end{aligned}$$

By comparing Equation 2.9 with the factoring of two linear equations, we can solve for the equation of each limb projection, namely

$$P_{2d_1}^{con} : y = \frac{\sqrt{m^2 - 4kl} - m}{2l} x + \frac{-\sqrt{o^2 - 4lp} - o}{2l} \quad (2.10)$$

$$P_{2d_2}^{con} : y = \frac{-\sqrt{m^2 - 4kl} - m}{2l} x + \frac{\sqrt{o^2 - 4lp} - o}{2l} \quad (2.11)$$

Two Planar Surfaces

Our equation of planar surfaces has three parameters (a, b, d) and takes the form

$$P_{3d}^{pln}(x, y, z; a, b, d) = ax + by + z + d = 0 \quad (2.12)$$

To find the equation of the projection of the crease edge between two intersecting planar surfaces, simply take two such equations and set them equal to each other.



$$P_{3d}^{pln_1}(x, y, z; a_1, b_1, d_1) = a_1 x + b_1 y + z + d_1 = 0$$

$$P_{3d}^{pln_2}(x, y, z; a_2, b_2, d_2) = a_2 x + b_2 y + z + d_2 = 0$$

$$P_{2d}^{pln+pln}(x, y; a_1, b_1, d_1, a_2, b_2, d_2) = (a_1 - a_2)x + (b_1 - b_2)y + (d_1 - d_2) = 0 \quad (2.13)$$

Note the resulting $P_{2d}^{pln+pln}$ equation is defined in terms of all six parameters that appear in $P_{3d}^{pln_1}$ and $P_{3d}^{pln_2}$. By fitting data points to P_{2d}^{pln} , we are in effect finding the parameters of the two intersecting planes.

Planar and Cylindrical Surfaces

Recall the expressions for P_{3d}^{pln} and P_{3d}^{cyl} from Equations 2.12 and 2.5. By solving for the variable z in P_{3d}^{pln} and substituting the expression into P_{3d}^{cyl} to eliminate the z term, we derive the equation of the projection of the crease edge created by the intersecting planar and cylindrical surfaces. Note that this equation has all the parameters that are present in P_{3d}^{pln} and P_{3d}^{cyl} .

$$P_{2d}^{pln+cyl}(x, y; a, b, d, x_0, y_0, r_0, \alpha, \beta) = ((y - y_0) \cos(\alpha) - (a x + b y + d) \sin(\alpha))^2 + ((x - x_0) \cos(\beta) + (y - y_0) \sin(\alpha) \sin(\beta) + (a x + b y + d) \cos(\alpha) \sin(\beta))^2 - r_0^2 \quad (2.14)$$

Planar and Spherical Surfaces

Using the same technique for deriving $P_{2d}^{pln+cyl}$, the equation of the projection of the crease edge created by two intersecting planar and spherical surfaces is

$$P_{2d}^{pln+sph}(x, y; a, b, d, x_0, y_0, z_0, r_0) = (x - x_0)^2 + (y - y_0)^2 + ((a x + b y + d) + z_0)^2 - r_0^2 \quad (2.15)$$

Other Surface Intersections

The 2D equations of the projection of intersection of other surface types can be derived similarly. However, they may involve very long expressions and are thus omitted here.

2.6 Registered Fused Images

A database of more than 40 registered (fused) range and intensity images has been created for experiments. The fused data is dense except for regions of shadowing and for small holes due to quantization. Each pixel in the fused image contains both the 3D positional information and the intensity value at that position. We refrain from creating a new terminology such as “fuse1” to denote the rich information at each image grid point. Instead, the term “pixel” will be used throughout this thesis and should be understood to contain both range and intensity data. The fused images have been acquired via the White Scanner from Technical Arts [117]. A brief discussion of the fusion process is given here. Refer to Appendix A for complete details of the fusion procedure.

The White Scanner is a triangulating sensor which projects a sheet of laser light onto objects which are viewed by a camera from another angle (see Figure 2.6). Normally, the intensity image from the camera is only used as intermediate data toward the computation of 3D coordinates in the workspace reference frame. We registered the intensity image pixels from the camera with range values obtained by parallel projecting computed 3D surface points along the optical axis of the camera onto a synthetic *fused image plane*. The *fused image plane* is a synthetic image plane that is coplanar with the CCD array of the camera and contains both a range value and an intensity value at each quantized point. A perspective transformation was obtained by calibration to map 3D world surface points onto the intensity array produced by

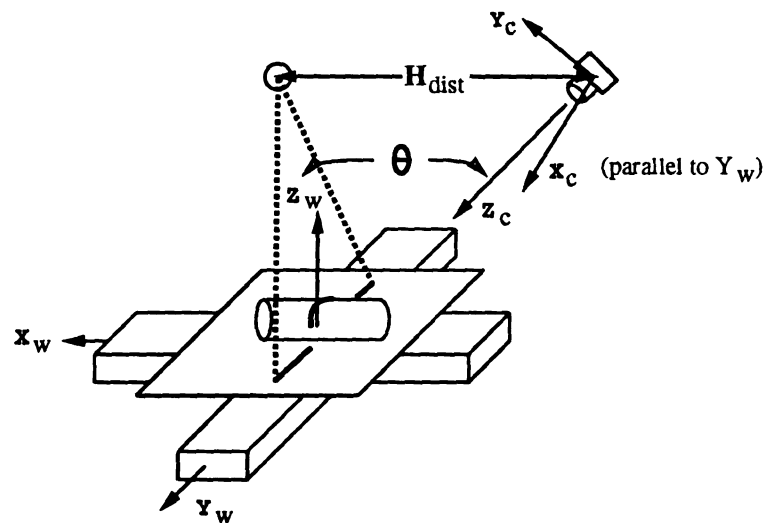


Figure 2.6. Setup of White Scanner and the two coordinate frames

the camera. In this manner, each visible surface point could be assigned an intensity in the fused image plane. Care was taken to assure registration accuracy to within one pixel in the final fused images. Because the White Scanner is a triangulating sensor, there are voids in the data due to shadowing from both the camera and laser sheet. Shadowing from the white light sources has been minimized by using two or three light sources when taking the intensity images.

One final note: because the laser illumination is not the same as the white light, it may happen that intensity edges and range jumps at limb boundaries are slightly out of registration.

2.7 Summary

In this chapter, we have defined the *wing primitive*. The structure of a wing and its mathematical representation were discussed. Essentially, a *wing* is a $2\frac{1}{2}$ D fragment of image contour along with knowledge about the surfaces that project to the regions

adjoining the image contour. A wing captures not only the shape of the object in the local region but also gives clues to the pose of the surfaces that form the wing. Given our definition of contour and surface labels, a set of 98 wing primitives were conceived. The wing detection algorithms of the next chapter detect wings that can be labeled by those wing primitives.

The *line drawing graph* is the assumed starting point and input of many past line drawing analysis research projects [30, 61, 66, 83, 115, 123]. However, the *labeled line drawing graph* defined here is much richer in information than the usual labeled line drawing. A LLDG contains not only the usual line interpretations (labels) but also contains information about the surfaces that project onto the regions in the graph. With this additional surface information, a higher level vision process may be able to determine the pose of the objects in the scene.

Finally, we briefly discussed the set-up in which the test images are generated. We outlined how one can fuse the range and intensity images, originally in different coordinate systems, to produce a single registered image. The details of the fusion process can be found in Appendix A.

CHAPTER 3

Wing Detection

3.1 Introduction

As with any representation theory for object recognition, successful extraction of scene features prescribed by the theory is as important as the theory itself. For theory of wing representation, the features to be extracted are *wings*. A list of wing primitives that can appear in the image are defined in Chapter 2. This chapter presents an approach to wing detection.

Instead of detecting the surface and contour components of the wing independently, we propose to extract, both wing contour and wing surfaces simultaneously. Hypotheses of presence of various wing primitives are first generated. Verification of each hypothesis is cast into a non-linear optimization problem in which convergence to a solution approves the hypothesis and also recovers the parameters of the wing. The verification procedure involves a novel idea of fused fitting of surface and image contour data simultaneously to recover the wing parameters. The fused fitting process is formulated and carefully studied in this chapter. Monte Carlo experiments are performed to test the quality of the estimated wing surface parameters from fitting with fused, surface-only and contour-only data and to test the shape discrimination capability of fused and surface-only fitting techniques. Experiments are also conducted

to study the minimum number of data points required to obtain quality parameter estimation. Finally, we report on results and inferences drawn from detection of wings on real fused images.

3.2 Survey of Related Background

The wing, as defined in Chapter 2, consists of a fragment of edge image contour and adjacent surface information. Using commonly known vision techniques, wing detection can be decomposed into two steps: edge detection followed by surface fitting for recovery of the surface properties and the pose information. Chen [26] experimented with the above simple strategy using only range images. Image segmentation was performed using techniques of either Hoffman & Jain [54], Laurendeau & Paussart [78], or Marr & Hildreth [85], afterwhich, a symbolic uniform procedure repeatedly grouped pixels into the most likely neighboring region. Another approach proposed by Chen was to perform feature extraction by means of the directional derivatives. Directional derivatives are first order derivatives of a point in a given direction. However, our implementation shows that a better wing detector is still needed. We should note that the wing as defined by Chen is a less specific than our wing definition (see Figure 2.5). A major difference is that pose information of the surfaces is embedded in our wing while Chen's has only qualitative surface labels.

Surface fitting has traditionally been done by fine tuning the parameters of an underlying assumed surface model and edge constraints are usually not used. However, edge information from objects provides strong clues about the local shape of surfaces [70]. For example, Lowe used only 2D contours for recognition of objects in general position [80]; furthermore, superquadric surface fitting with edge constraints has been successfully demonstrated by Bajcsy and Solina [5]. More recently, Lowe [81], Taubin [116], and Bolle *et al.* [17] have also demonstrated the importance of contour



information by utilizing it in estimating initial parameters prior to fitting and/or using it to aid the recovery of parameters during fitting. Our wing detector explicitly uses the contour information to help deduce the initial parameter estimates and guide the non-linear optimization process.

The following subsections survey some of the traditional edge detecting, surface segmenting and surface fitting techniques.

3.2.1 Edge Detection in Intensity Imagery

The work on edge detection in intensity imagery can be traced back to the 2×2 cross operator of Roberts [108]. An approximate gradient is computed by convolving the edge operator with the image in two orthogonal directions and (non-)linearly combining the convolved values. The direction of the maximum gradient can also be approximated from the two convolved values. Since Roberts, many other different edge masks have been proposed, including the Sobel [33] and the Prewitt [102] operators. Instead of convolving the edge mask in only two directions and approximating the maximum gradient direction from it, better results can be achieved by convolving the edge operator in many different directions and selecting the maximum value obtained as the gradient magnitude at the given location, and the corresponding direction as the gradient direction. The Prewitt [102] and Kirch [69] are two such operators. However, the drawback of such compass gradient operators is that they require considerably more time for processing on a sequential computer.

Another type of window based operators, proposed by Hueckel [59, 60], Hummel [62], and Frei & Chen [45], fits an explicit model of the ideal edge/line to the image. Rather than fitting the model directly, the model is approximated by an orthogonal set of basis vectors (masks) so that existence and the orientation of the edge/line is computed by simple convolutions.

A well known and popular edge detector for intensity imagery is the Canny edge detector [22]. Canny defines detection and localization criteria for a class of edges and augments it with a criterion to ensure that the detector has only one response to a single edge. The major distinctions between his detector and the zero-crossing based detector [85] are that (1) the detector has only one response to a single edge, and (2) it uses directionally sensitive elongated (rather than circular) operators. Many post-Canny edge detectors have been published in the literature, but the Canny edge detector is still the norm against which all other intensity edge operators are compared. A brief summary of the recent research trends in edge detection is given in [20].

3.2.2 Segmentation of Range Imagery

In general, the intensity recorded at each pixel of an intensity image depends not only on the surface position and orientation but also on the surface material, light source position and the viewing position. Many researchers have concentrated on the recovery of explicit shape information from intensity image(s) (*e.g.*, shape-from-shading [56, 57], shape-from-occluding contours [122], shape-from-texture [126], and shape-from-stereopsis [8]) which are collectively called "shape-from-x". A survey of those techniques can be found in [1].

With rapid advances of technology, range sensors have become more affordable and accurate [10] and a new class of edge detectors for range images has evolved. The edge detector for intensity imagery can only be applied to range images to detect jump (depth discontinuity) edges. Crease edges cannot always be directly detected using those operators because of smooth transition in depth. As a remedy, Mitche and Aggarwal [87] offered a statistical test for detection of crease edges in range images. Different partitions at a neighborhood of interest were tested for evidence of a crease edge. A planar surface is fit to each partition and the difference between the slopes

of the two planar fits serves as the test statistic. Rather than using those intensity image operators, some pure range image segmentation methods have been proposed and are based on aggregation of local surface properties such as curvatures or surface normals [12, 35, 39, 54, 128]. Since curvature is a second order differential property, it is especially sensitive to noise. In [43], Flynn compared several different curvature estimation techniques but found no one performed satisfactorily under the influence of noise.

Among the many range segmentation techniques in the literature, there are two major classes: region-based (*e.g.*, [12, 54, 107]) and edge-based (*e.g.*, [35, 96, 99]). Region-based algorithms segment the image by finding homogeneous regions corresponding to surfaces of the objects while edge-based methods try to extract lines from images that correspond to boundaries of the surfaces. While a region-based algorithm may be poor in delimiting the region boundaries, the edge-based approach has the inherent drawback that the extracted line segments may be fragmented so that a heuristic line linking procedure is often required to ensure closed curves. Recently, Nadabar [90], Yokoya & Levine [128], Davignon [31], and Ferrie *et al.* [39] have given hybrid approaches to range image segmentation and have shown that by integrating the region-based and edge-based approaches and using their complementary information about the scene, a segmentation is achieved that has a better correspondence between the shape of the regions and the surface of the objects.

3.2.3 3D Surface Reconstruction: Superquadric and Quadric Fitting

Fitting of implicit quadric and superquadric forms to range data is one of the ways to reconstruct surface shape. Here we will concentrate on the fitting aspect of the reconstruction since our wing detector also involves quadric surface fitting.

The problem of 3D surface reconstruction has received enormous attention in vision research. A survey of recent 3D surface reconstruction methods can be found in Bolle [18]. For example, Rimey & Cohen [107] divide the range image into windows, classify each window as either planar, cylindrical or spherical and group like windows into surface regions. Grouping windows of the same surface types is cast as a weighted maximum likelihood clustering problem. The mixed windows, if any, are segmented using a maximum likelihood hierarchical segmentation algorithm.

Quadric surface representation, popularized by Faugeras [37], is to represent object surfaces by simple geometric quadric shapes such as a plane, sphere, cylinder, and cone. The implicit equation of a general quadric surface is given in Equation 2.1. By fitting this implicit equation to surface points to recover the coefficients, the underlying surface shape and its parameters can be estimated.

When fitting superquadrics in general position, 11 parameters need to be estimated. These are location (3), orientation (3), size (3) and shape (ϵ_1 and ϵ_2) parameters. Given the surface equations (see Section 1.2), the reconstruction proceeds as a nonlinear optimization procedure in which the error between the sampled surface points and the model is minimized. The most popular nonlinear optimization technique seems to be the Levenberg-Marquardt method [101]. As with most nonlinear optimization strategies, it requires an initial estimate of the model parameters and converges to the global optimum only if the initial estimate is already close to the global optimum. Much recent research on quadric and superquadric fitting only differs in the merit function for evaluating the goodness of fit and the estimation of the initial parameters. The well known research includes that of Pentland [98], Gross & Boulton [48], Ferrie *et al.* [38], Solina & Bajcsy [112], and Gupta & Bajcsy [50],

Recently, Kriegman & Ponce [73] proposed a method for shape reconstruction based on the rim contour points only. They use Elimination Theory to derive the implicit equation of the rim contour of surfaces of revolution and solve for the pa-

rameters by fitting a large number of contour points to the rim model using the Levenburg-Marquardt optimization technique. The implicit equation is a complicated high degree polynomial.

All of the fitting routines mentioned above assume that the image is pre-segmented and all use a large number of surface points sampled over a large surface area for fitting (in [73], it is contour points over most of the rim contours.). The large surface (contour) patch requirement is needed for better estimation of the initial parameters. As we will show later in this chapter, reconstruction of surface shape can be done with few sample points over a small surface area if the boundary information can be used during the fitting process. Moreover, images do not have to be pre-segmented. As already pointed out at beginning of this section, image contour information can be an important cue in extracting surface shape. In the remainder of this chapter, we will explore the possibility of simultaneous fitting 3D surface data points together with 2D image contour points.

3.3 Wing Detection Algorithm

A flowchart of the proposed approach is depicted in Figure 3.1. Instead of detecting all wings in the input image I at once, we propose to divide the input image I into overlapping windows and to detect wings in each of those windows in parallel. Since wings are local features, locating all wings at once from the original image is recast into locating wings from smaller subimages that overlap the original image. The set of detected wings may contain many overlapping and co-curvilinear wings, in which case, they may be merged to form longer wings and to achieve better accuracy for the wing parameters. The functionality of each module of the algorithm is examined more closely in the following subsections and a list of the major parameters of the algorithm can be found in Appendix C.

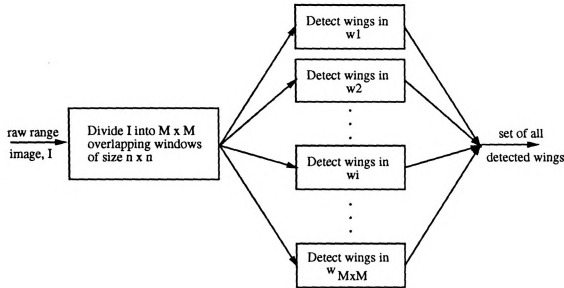


Figure 3.1. Flowchart of proposed wing detection algorithm

3.3.1 Tessellation of the Input Image

The first step in the wing detection process is to divide the input image $I_{N \times N}$ into $M \times M$ overlapping windows, $w_1, w_2, \dots, w_{M \times M}$, of size $n \times n$ (see Figure 3.2) such that

1. $\bigcup_i w_i = I_{N \times N}$, and
2. each pixel $P_{r,c}$, except for those that are near the image border, is covered by exactly m windows.

The constants, $\{n, m, N, M\}$ are certainly all interrelated. The number of windows, $M \times M$, is directly related to the window size, n and the degree of overlapping, m . The size of the window is crucial to the success of wing detection. If the window is too small, then the covered surface patch would also be small, in which case, the number of data points available for the parametric fitting may be too few to be accurate. Worse yet, with a small surface patch, it may be impossible to derive a good set of starting values for the non-linear fit. On the other hand, large window size defeats the purpose of dividing the original image into smaller size to simplify the wing

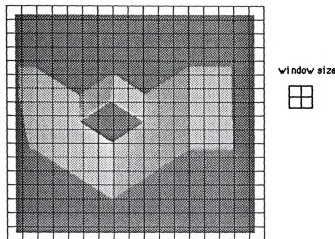


Figure 3.2. Tesselation of input image

detection process. In general, there is no one window size that will work well for all images. The best set of $\{n, m, N, M\}$ to use is really problem, feature size and pixel size dependent. With larger features that expand across a large area of the image, a large window would not hurt the detection of features. However, small features may be missed by using large windows. Given the drawback of a fixed window size, a scale space approach in wing detection using coarse to fine window sizes is an important future research direction.

The reason for overlapping windows instead of disjoint segmentation is to guard against the fact that wings may fall on the border of a window, in which case, the wing may not be detected. By having overlapping windows, we try to minimize the chance of a wing falling on borders of all windows that (partially) cover it.

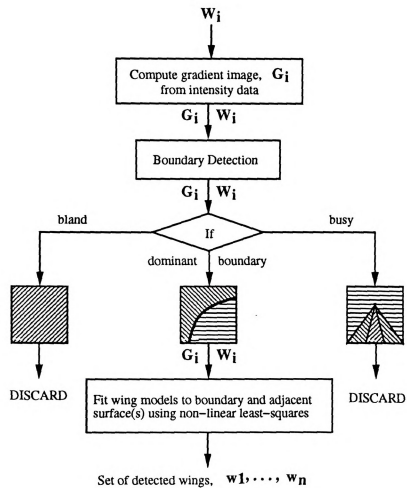
3.3.2 Detecting Presence of Wings

A hierarchical diagram of how wings may be detected from each subimage is depicted in Figure 3.3.



In the first stage, we screen out those windows that are not interesting or are too busy and only keep those windows that are moderately busy for wing detection. Since a wing contains a fragment of an edge contour, if a window covers only the image of a single surface patch, there can be no wing in that window and no wing detection is necessary in that window. We call those windows *bland* windows. On the other hand, windows that are too *busy* will also be discarded. A window is said to be *busy* if more than one wing is present. A typical busy window would be one in the neighborhood of a vertex. It has been shown in the literature that edge detection near a corner is a difficult task. Many have attempted to find corners or dominant points on planar curves [4, 36, 44, 106, 110, 118], which would require the image to be segmented in the first place. By detecting wings away from busy areas of the image and later reconstructing the junctions from the detected wings, the true junction type and location might be better estimated. Thus, only those windows that contain exactly two surface patches, counting the background as one, will be further processed. Those are the *interesting* windows.

A simple procedure is used to detect the presence of wing(s) and to identify those interesting windows. First the intensity map of the subimage is convolved with the Sobol operator. The $2 \times w_{size}$ pixels with the highest gradients from the above convolution are selected as potential edgelets. From the definition of wings in Chapter 2, there are three classes of wing contours: linear, circular and parabolic. That is, the 3D edge when projected onto the image plane can be locally approximated by either a straight line, a circle or a parabola. Therefore, the decision criterion for an interesting window is based on the goodness of fit of those wing contour models to the potential edgelets. Note that those 2D curve models have 2, 3, and 3 parameters. By selecting enough edgelet points, we can solve for the parameters of the curve models using the linear least squares method. If the window is bland or contains many 3D edges, then all these fits should be poor and the window will be rejected. Otherwise,

Figure 3.3. Wing detection within each subimage w_i

in an effort to remove outliers and to achieve more accurate wing location, those potential edgelets that deviate from the fit model by more than the mean deviation of all potential edgelets are thrown out as outliers. The remaining edgelets are fit to the same model again to obtain the final wing contour. The equation of the fitted curve is useful in the estimation of the initial parameter values for the iterative non-linear fitting to be described in Section 3.4. Furthermore, the policy for sampling range data for wing model fitting can also be based on the shape of the contour extracted and the location of the contour.

Note that the selection of the edgelets is based on the Sobel gradient of the intensity image; the range data is not used. We are able to successfully extract most edgelets using this simple procedure. Obtaining edgelets without the use of range data also paves the way for our future research on wing detection from intensity image alone.

3.3.3 Fitting Wing Primitives

Once interesting windows have been identified, the next step is to detect plausible wings from each of these windows. This is achieved by fitting each of the wing models to a sub-sample of points from each of the two regions and the edge contour in the window. As pointed out in Chapter 2, if a wing model assumes a jump edge then the 2 surfaces are fit individually without the benefit of contour data. Otherwise, either one or both surfaces may be fit with the aid of the contour points. We follow the popular technique of Levenberg-Marquardt for fitting non-linear models [101]. A non-linear model depends nonlinearly on the set of unknown parameters that define it. A χ^2 merit function is defined and the best-fit parameters are determined by minimization of the merit function. If no wing model fits the sampled points well, then no wing is detected in that window. Otherwise, all detected wings from a window are reported back for higher level processing.

In the next section, we will develop the details of the fitting equation, the χ^2 merit function and the initial estimate of the wing model parameters.

3.4 Simultaneous Boundary and Surface Fitting

Least-squares fitting has been one of the most popular fitting techniques in the field of computer vision [16, 48, 68, 73, 81, 100, 112, 116]. The conceptual idea is as follows. Given a set of n data points $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$, we want to fit the data by an implicit equation, $P(\mathbf{x}_i; \Omega) = 0$, having parameters Ω . The problem is to find a set of parameter values $\hat{\Omega}$ such that $P(\hat{\Omega})$ models those data points well. $\hat{\Omega}$ is said to be the least-squares estimate (LSE) of the parameters that minimizes the sum of squared errors. Thus,

$$\sum_{i=1}^n (P(\mathbf{x}_i; \hat{\Omega}))^2 \leq \sum_{i=1}^n (P(\mathbf{x}_i; \tilde{\Omega}))^2 \text{ for all possible parameter values } \tilde{\Omega}.$$

It is well known that the residual error measure may be a poor indicator of the error between the data points and the fitted model [76]. For example, when fitting a polynomial to a set of points in 2D, points that have the same Euclidean distance to the fitted polynomial may have grossly different residual errors (see Figure 3.4).

A better approach is to use the exact normal distance function when computing fitting errors. Instead of minimizing the squared residual errors, the sum of the squared normal distances between the data points and the fitted curve is minimized. Hence the quantity to be minimized is

$$\sum_{i=1}^n \text{dist}(\mathbf{x}_i, P(\mathbf{x}_i; \hat{\Omega}))^2.$$

Unfortunately, the exact formulation of the distance between the data points and the fitted curve often does not exist (*e.g.*, determining distance between a 3D point

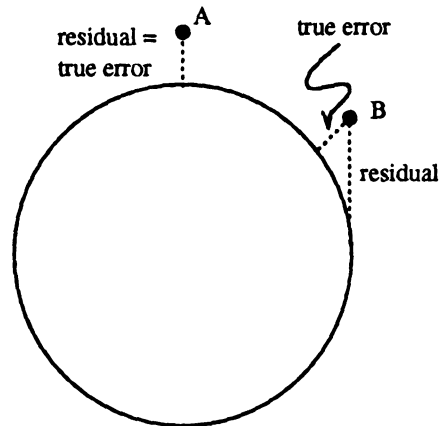


Figure 3.4. Data points that are close to the fitted curve may have large residual errors.

and a general quadric surface). Some surface fitting algorithms in the computer vision community have either continued to use the residual errors [42] or used an approximation to the true distance function [16, 68, 73, 112, 116]. The study by Gross and Boulton [48] showed that the *ad hoc* goodness-of-fits measures have many problems and biases in parameter fitting. It was shown that fitting superquadrics with an error measure based on the true Euclidean distance fares far better than if other *ad hoc* measures are used.

In what follows, we will define our fitting functions and derive the true distance measure to be used as the indicator for the goodness of the fit. We are able to derive the true distance measure because explicit quadric surface models are used. This would not be possible if a general quadric surface equation were used instead.

For convenience, denote the candidate window for wing detection as W_i . Instead of using all the points in the window, we subsample m surface points from each region and n contour points along the wing contour as data points for wing fitting. Let's denote those points as $\mathcal{X}_{s1} = \{S_{11}, S_{12}, S_{13}, \dots, S_{1m}\}$, $\mathcal{X}_{s2} = \{S_{21}, S_{22}, S_{23}, \dots, S_{2m}\}$, and $\mathcal{X}_c = \{C_1, C_2, C_3, \dots, C_n\}$, respectively. The effect of different sizes of m and n have been studied and is reported in Section 3.6.



3.4.1 Fitting Equations

There are 3 different fitting functions used depending on the type of the wing contour of the assumed wing model. If the wing contour of the assumed wing model is of type convex or concave crease, then the fitting function is

$$F_1(\mathbf{x}_i; \Omega) = w_1 (\mathbf{x}_i \in \mathcal{X}_{s1}) P_{3d}^{s1}(\mathbf{x}_i; \Omega) + w_2 (\mathbf{x}_i \in \mathcal{X}_c) P_{2d}(\mathbf{x}_i; \Omega) + w_1 (\mathbf{x}_i \in \mathcal{X}_{s2}) P_{3d}^{s2}(\mathbf{x}_i; \Omega) , \quad (3.1)$$

for all points in $\mathcal{X} = \mathcal{X}_{s1} \cup \mathcal{X}_c \cup \mathcal{X}_{s2}$. The expressions $(\mathbf{x}_i \in \mathcal{X}_{s1})$, $(\mathbf{x}_i \in \mathcal{X}_c)$, and $(\mathbf{x}_i \in \mathcal{X}_{s2})$ are binary terms. For any \mathbf{x}_i , only one of the terms is 1 (true) and the other two are 0 (false). The weights w_1 and w_2 are used to scale the relative significance of image contour residual compared to range residual. Note that there is also an implicit weighting caused by the numbers of surface points m and contour points n .

If the wing contour is of type “jump”, then contour information is not useful in the recovery of the surface parameters. In this case, the wing is broken into two half-wings and each half-wing is fit to only the set of sampled surface points. Thus, across jump edges the fitting function is the same as the surface model equation, which is

$$F_2(\mathbf{x}_i; \Omega) = P_{3d}^{s1}(\mathbf{x}_i; \Omega), \forall \mathbf{x}_i \in \mathcal{X}_{s1}$$

and

$$F_2(\mathbf{x}_i; \Omega) = P_{3d}^{s2}(\mathbf{x}_i; \Omega), \forall \mathbf{x}_i \in \mathcal{X}_{s2}. \quad (3.2)$$

If the wing contour of the assumed wing model is of type limb, then the contour data can only aid the recovery of the self-occluding surface. The fitting equation for this surface and the contour is

$$F_3(\mathbf{x}_i; \Omega) = w_1 (\mathbf{x}_i \in \mathcal{X}_{s1}) P_{3d}^{s1}(\mathbf{x}_i; \Omega) + w_2 (\mathbf{x}_i \in \mathcal{X}_c) P_{2d}(\mathbf{x}_i; \Omega), \quad (3.3)$$

for all $\mathbf{x}_i \in (\mathcal{X} = \mathcal{X}_{s1} \cup \mathcal{X}_c)$. Since a limb edge is a jump edge, the other surface which is being occluded by the self-occluding surface is detected without the aid of the contour information. Its fitting equation would be the same as Equation 3.2.

3.4.2 Distance Functions

If we substitute P_{3d}^{s1} and P_{3d}^{s2} in the fitting equations with one of $\{P_{3d}^{pln}, P_{3d}^{sph}, P_{3d}^{cyl}, P_{3d}^{con}\}$ and P_{2d} with one of $\{P_{2d}^{pln}, P_{2d}^{sph}, P_{2d}^{cyl}, P_{2d}^{con}\}$ and do a least-squared fit over F_1, F_2 or F_3 then we would be computing the LSE of the parameters by minimizing the squared residual errors. As Figure 3.4 indicates, a better error function is the true distance function. Since we have modeled different types of quadric surfaces by their respective explicit quadric equations, we are able to derive the true distance from a given 3D point, \mathbf{x}_{3d} , to the model surface. The distance function from a given 2D contour point, \mathbf{x}_{2d} to the model 2D contour may also be derived. We shall denote the distance function as DP_{3d}^{qqq} and DP_{2d}^{qqq} where qqq denotes one of the surface/contour types.

Planar. The way P_{3d}^{pln} is defined, the distance from \mathbf{x}_{3d} to the fitted plane is equal to the residual at that point. Thus the distance function of P_{3d}^{pln} is simply $|P_{3d}^{pln}|$. Computation of the distance from \mathbf{x}_{2d} to a line in the same image plane is also straight forward. If the line has slope m and y -intercept b , then the distance is simply $(mx + y)/\sqrt{1 + m^2}$. Thus,

$$\begin{aligned}
DP_{3d}^{pln_1}(x, y, z; a_1, b_1, d_1) &= |a_1x + b_1y + z + d_1|, \\
DP_{3d}^{pln_2}(x, y, z; a_2, b_2, d_2) &= |a_2x + b_2y + z + d_2|, \\
DP_{3d}^{pln+pln}(x, y, z; a_1, b_1, d_1, a_2, b_2, d_2) &= \left| \frac{P_{2d}^{pln+pln}}{\sqrt{1 + \frac{(a_2-a_1)^2}{(b_1-b_2)^2}}} \right|.
\end{aligned}$$

Sphere. Computation of the distance from \mathbf{x}_{3d} to a fitted sphere can be computed as the distance from \mathbf{x}_{3d} to the center of the sphere minus the radius of the sphere. In 2D, the distance from \mathbf{x}_{2d} to the limb contour projection is the distance from \mathbf{x}_{2d} to center of the projected circle minus the radius of the circle. Thus,

$$\begin{aligned}
DP_{3d}^{sph}(x, y, z; x_0, y_0, z_0, r_0) &= |\sqrt{(x-x_0)^2 + (y-y_0)^2 + (z-z_0)^2} - r_0|, \\
DP_{2d}^{sph}(x, y; x_0, y_0, z_0, r_0) &= |\sqrt{(x-x_0)^2 + (y-y_0)^2} - r_0|.
\end{aligned}$$

Cylindrical. The distance from \mathbf{x}_{3d} to a cylindrical surface is also straight forward. The distance is the absolute difference between the radius of the cylinder and the distance from \mathbf{x}_{3d} to the axis of the cylinder which is a straight line in 3D. Computation of the distance from \mathbf{x}_{2d} to the projected limb contour is similar to that of $DP_{2d}^{pln+pln}$. Note that the slope of the projected limb contour in P_{2d}^{cyl} is $\cot(\beta) \sin(\alpha)$.

$$\begin{aligned}
DP_{3d}^{cyl}(x, y, z; x_0, y_0, r_0, \alpha, \beta) &= \\
& (y_0 - y) \cos(\alpha) \sin(\alpha) \cos(\beta)^2 + (x - x_0) \cos(\alpha) \cos(\beta) \sin(\beta) - \\
& \left(\frac{1}{(\sin(\alpha)^2 + \cos(\alpha)^2 \sin(\beta)^2)} \right) ((\sin(\alpha)^2 + \cos(\alpha)^2 \sin(\beta)^2) r^2 + \\
& 2(-x^2 + 2xx_0 - x_0^2) \cos(\alpha)^2 \cos(\beta) \sin(\alpha) \sin(\beta) + \\
& 2(-xy + x_0y + xy_0 - x_0y_0) \sin(\alpha) \sin(\beta) \cos(\beta) + \\
& (-y^2 + 2yy_0 - y_0^2) (\cos(\alpha)^4 + \sin(\alpha)^4) \sin(\beta)^2 + \\
& 2(-y^2 + 2yy_0 - y_0^2) \cos(\alpha)^2 \sin(\alpha)^2 \sin(\beta)^2) \\
& - z,
\end{aligned}$$

$$DP_{2d}^{cyl}(x, y, z; x_0, y_0, r_0, \alpha, \beta) = \left| \frac{P_{2d}^{cyl}}{\sqrt{1 + \cot^2(\beta) \sin^2(\alpha)}} \right|.$$



Conical. We use Figure 3.5 to explain how the exact distance function from \mathbf{x}_{3d} to the surface of a cone can be derived. The distance from \mathbf{x}_{3d} to the conical surface is merely

$$\text{dist}(\mathbf{x}_{3d}, P_{3d}^{con}) = l_1 \cdot \sin(\theta_2).$$

By the Law of Sines, θ_2 can be computed as $\sin^{-1}(\frac{l_2}{l_1}) - \theta_1$, where l_1 is the distance between \mathbf{x}_{3d} and the vertex, and l_2 is the distance between \mathbf{x}_{3d} and the axis of the cone. Both quantities are easily computed given the estimated parameters $\hat{\Omega} = \{x_0, y_0, z_0, d_0, \alpha, \beta\}$ of the cone. θ_1 , which is the angle between the axis and the surface of the cone, is just $1/d_0$.

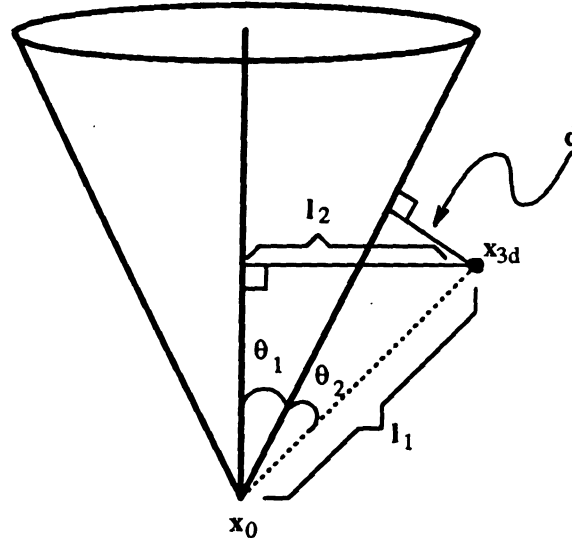
The formulation for computing the distance from \mathbf{x}_{2d} to the limb contour of the cone is similar to the cylindrical case. As with the cylindrical case, since the 2D equations of those limb contours are already derived (Equations 2.10 and 2.11), one only needs to find the distance between a point and a line.

$$\begin{aligned} DP_{3d}^{con}(x, y, z; x_0, y_0, z_0, d_0, \alpha, \beta) &= l_1 \cdot \sin(\theta_2) \\ DP_{2d}^{con}(x, y, z; x_0, y_0, z_0, d_0, \alpha, \beta) &= \left| \frac{P_{2d}^{con}}{\sqrt{1 + \text{slope}^2}} \right|, \end{aligned}$$

where *slope* is the slope of the line P_{2d}^{con} (refer to Equations 2.10 and 2.11 for exact notation).

3.4.3 χ^2 Merit Functions

Following [101], we will use the χ^2 merit function to evaluate the goodness-of-fit. Assume that each surface data point, $\mathbf{x}_i \in \mathcal{X}_{s1}$ or $\mathbf{x}_i \in \mathcal{X}_{s2}$, has a measurement error that is independently and identically distributed as a normal distribution around the fitting model $P_{3d}(\mathbf{x}_i; \Omega)$, with zero mean and standard deviation σ_s . Likewise, assume that each contour data point, $\mathbf{x}_i \in \mathcal{X}_c$, has a measurement error that is also independently and identically distributed as a normal distribution around the contour model $P_{2d}(\mathbf{x}_i; \Omega)$ with zero mean but different standard deviation σ_c (i.e.,



Given: θ_1 , x_{3d} , x_0

$$l_1 = \text{DIST}(x_{3d}, x_0)$$

$$l_2 = \text{DIST}(x_{3d}, \text{axis of the cone})$$

$$\theta_2 = \sin^{-1}\left(\frac{l_2}{l_1}\right) - \theta_1$$

$$d = l_1 * \sin(\theta_2)$$

Figure 3.5. Computing the Euclidean distance from a point in 3D to the surface of a cone.

$DP_{3d}^{surface}(x_{3d}, P_{3d}^{surface}) \sim N(0, \sigma_s)$ and $DP_{2d}^{contour}(x_{2d}, P_{2d}^{contour}) \sim N(0, \sigma_c)$. Then the χ^2 merit functions for the the fitting functions defined earlier are

$$\chi^2(\Omega) \equiv \begin{cases} \sum_{i=1}^{2m+n} \left(\frac{F_1(\mathbf{x}_i; \Omega)}{\sigma_1}\right)^2, & \text{for fitting } F_1 \text{ in Equations 3.1 (creases)} \\ \sum_{i=1}^m \left(\frac{F_2(\mathbf{x}_i; \Omega)}{\sigma_2}\right)^2, & \text{for fitting } F_2 \text{ in Equations 3.2 (jumps)} \\ \sum_{i=1}^{m+n} \left(\frac{F_3(\mathbf{x}_i; \Omega)}{\sigma_3}\right)^2, & \text{for fitting } F_3 \text{ in Equations 3.3 (limbs)} \end{cases} \quad (3.4)$$

For F_2 in Equation 3.2 where only one surface model is involved, σ_2 is simply σ_s . σ_1 and σ_3 are derived from the sum of two χ^2 s. Their derivations are given in Appendix B.

By minimizing the χ^2 quantity, we obtain the maximum likelihood estimate of the model parameters if the noise follows the distribution assumptions. Note that we have substituted DP_{3d}^{**} for P_{3d}^{**} and DP_{2d}^{**} for P_{2d}^{**} in the fitting functions so that the least-squares estimates are computed in terms of true distance errors instead of residual errors.

3.4.4 Initial Parameters

With an iterative non-linear optimization technique, we must supply the fitter with an initial estimate of the parameters. The fitter iteratively refines the supplied initial values until convergence. The final estimated parameters are more likely to be the global minimum if the initial estimates are close to it in the first place. Thus it is important to have good initial estimate of the parameters. In [80], Lowe matched features of the hypothesized model to features found in the image to help estimate the pose parameters of the hypothesized object. Using this paradigm, not only are the model database must be available, the challenging problem of matching image features to model features must also be solved. Furthermore, there could be large number of matches if only a small number of features were detected. However, Lowe claimed that if the initial orientation parameters are within 60 degrees of the correct values, then almost any values can be chosen for the other parameters. Our approach, which will be described in the following subsections, is to estimate the parameters from the small surface patch and make good use of the contour information whenever it is available.

Planar. The parameters of a plane are completely recovered if the surface normal and a point on the plane are known. The surface normal can be computed by taking the cross product of any two non-collinear vectors on the plane. In our implementation, we picked three non-collinear surface points that are far apart in the region and use two of the three surface vectors formed by those three points to compute the surface normal. The normal vector along with any of the three points picked uniquely determines the plane on which the three points lie.



Spherical. A spherical surface model has four parameters, the center of the sphere (x_0, y_0, z_0) and the radius of the sphere r_0 . There are two cases to be considered when the assumed surface model is spherical.

Case 1: Wing contour is of class “circular” and of type “limb”. Note that the orthographic projection of the limb boundary of the sphere onto the image plane (the $z = 0$ plane) is a circle whose radius is the same as the sphere. Furthermore, since the viewing direction is along the z -axis, the center of the sphere must lie on the line parallel to the z -axis and passing through the center of the circle in 2D. This means the parameters x_0 and y_0 can also be extracted from the equation of the limb on the image plane. With an additional point on the spherical surface patch, say (x_i, y_i, z_i) , the lone remaining parameter, z_0 , can be computed by solving the equation of the sphere for z_0 . Thus,

$$z_0 = z_i + \sqrt{r_0^2 - (x_i - x_0)^2 - (y_i - y_0)^2}$$

Since the equation of the projected circle on which the wing contour lies is already known from the wing presence test in Section 3.3.2, all of the parameters can be estimated using the procedure outlined above.

Case 2: Estimating parameters from surface patch. Without information about the limb, the parameters must be estimated from the surface patch directly. Instead of estimating the curvatures of surface points, which are second derivative features and therefore are noise sensitive, and deriving the parameters from the estimated curvatures as in [42], we simply fit the surface points with the implicit equation of a sphere. This implicit equation has the form:

$$P_{3d}^{sph}(x, y, z; a, b, c, d) = x^2 + y^2 + z^2 - ax - by - cz - d = 0 \quad (3.5)$$

The parameters of the spherical surface model can then be computed from the parameters of the implicit equation $\{a, b, c, d\}$ by

$$x_0 = a/2, \quad y_0 = b/2, \quad z_0 = c/2, \quad r_0 = \sqrt{d - \left(\frac{a}{2}\right)^2 - \left(\frac{b}{2}\right)^2 - \left(\frac{c}{2}\right)^2}.$$

Cylindrical. There are two translational and two rotational parameters along with the radius of the cylinder to be estimated. In effect, one only need to recover the radius and the orientation of the axis in 3D to estimate all five parameters. Again, there are two cases to be considered when the assumed wing surface model is cylindrical.

Case 1: Wing contour is of class "linear" and of type "limb". As in the spherical case, the presence of the limb contour is useful in estimating the initial parameter values. However, the limb contour information is not used in estimating the radius of the cylinder. Although the limb contour gives the direction of the minimum curvature of the surface patch in 2D, the maximum curvature (therefore the radius) cannot be computed directly because the direction of the maximum curvature is not along a straight line in the 2D image. Instead, the radius of the cylinder is estimated by first fitting a bi-quadratic surface of the form

$$z = P(x, y; a, b, c, d, e, f) = ax^2 + by^2 + cxy + dx + ey + f$$

to sampled surface points from the surface patch and then computing the maximum curvature at each of the sampled points. The radius of the cylinder is taken to be the inverse of the median of all the maximum curvatures.

The axis of the cylinder must lie parallel to the wing contour in the image plane. Furthermore, any straight line on the surface patch that is parallel to the wing contour in the image plane must have the same orientation as the axis of the cylinder. Hence, by shifting the line along the wing contour into the region corresponding to the



cylindrical surface patch, the surface points that lie on the new line form a 3D line that has the same orientation as the axis of the cylinder. The rotational angles can now be estimated from the orientation of this 3D line. Finally, the translational parameters are estimated by first projecting this 3D line into the cylinder along the surface normal by a distance equal to the estimated r_0 . The new 3D line that goes through those projected points is an approximation to the axis of the cylinder. The translational parameters can then be determined by extending the axis to intersect the image plane (the xy -plane).

Case 2: Estimating parameters from surface patch. The radius is estimated using the same procedure as in Case 1. Since the orientation of the cylinder in the 2D image plane is not known in this case, we project each of the sampled surface points along its surface normal into the cylinder by a distance equal to the estimated r_0 . Those projected points should lie roughly on a line which is the axis of the cylinder, from which the translational and rotational parameters of the cylinder can all be estimated.

In Section 3.7 the cylindrical parameters estimated in Case 1 result in better estimates due to the fix on the axis direction in 2D.

Conical. Although the fitting equations have been derived, we are unable to derive a satisfactory algorithm for estimating the geometric parameters of the conical model from a small conical surface patch, even if information about a limb boundary is available. We have left the problem of estimating conical parameters from a *small* surface patch to future research. This is not to say that there is no hope in initial parameter estimation for conical models. In fact, several papers [17, 42, 95] addressing quadric surface fitting have reported methods in estimating those parameters. However they all have their limitations. Often, a large surface patch is assumed and/or they use

noise-sensitive second derivative features such as curvatures to derive the initial values. For example, Flynn and Jain [42] proposed a method for estimating conical model parameters based on curvature values. A bicubic surface is fit to the surface patch, from which the maximum and minimum curvatures and their directions are estimated. The geometric parameters of the conical models are computed based on those curvature values. As the authors pointed out, these calculations require accurate curvature estimates, which in itself is a very difficult task [43]. Also, no examples of conical parameter estimation were given. In [95] a model driven technique for estimating conical surface parameters was given. This method is model driven because one of the parameters, namely the angle between the axis and the cone surface, is assumed known from the model database. The other parameters, the vertex of the cone and axis direction, are inferred from the surface patch which must include both limbs. However, from the outlined approach and the examples given in the paper, it seems that a large surface patch size is required to obtain acceptable initial parameter values. Bolle [17] approximated the axis direction of quadrics of revolution by intersecting two planes in 3D that theoretically contain the axis of the cone. Each plane was chosen to contain a surface point and spanned by the direction vectors of minimum curvature and normal at that point. We think that this approach merits some consideration in our future work. By using information about the axis direction in 3D and the limb contour in 2D, the initial estimates of the conical model can be defined.

Our work so far shows that it is very difficult to differentiate a cone from a cylinder when only one limb is present. It may be that we should consider them locally equivalent.

3.4.5 An Example

Figure 3.6 depicts how two wing models are fit to a window that partially covers one side of a cylinder including the limb boundary and the background. In this window, since only one surface exists, the wing is detected using half-wing models. The background requires no fitting at all. In Case 1, where the correct wing model ($NIL + LN_{<<} + CYL$) is to be fitted, the half-wing used is ($CYL + LN_{<<}$). Since the wing contour is assumed to be the projection of a limb boundary, both surface and contour points are fitted simultaneously. Hence, the fitting function is

$$F(\mathbf{x}_i; \Omega) = w_1 (\mathbf{x}_i \in \mathcal{X}_s) P_{3d}^{cyl}(\mathbf{x}_i; \Omega) + w_2 (\mathbf{x}_i \in \mathcal{X}_c) P_{2d}^{cyl}(\mathbf{x}_i; \Omega),$$

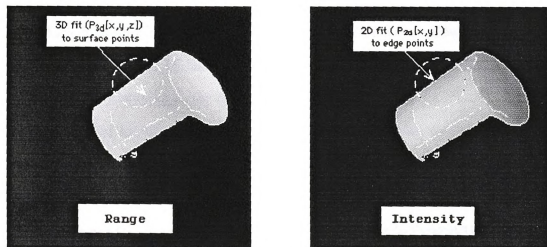
where $\mathbf{x}_i \in \mathcal{X}_s \cup \mathcal{X}_c$.

In Case 2, an incorrect underlying wing model ($NIL + LN_{<} + CYL$) is assumed. Since the wing contour is assumed to be the projection of a jump edge, only the sampled surface points are used for fitting. Note that the assumed surface model is correct so that the fit may converge. However, it is our experience that initial parameter estimation is more difficult and less accurate and therefore less likely to converge. The fitting function is,

$$F(\mathbf{x}_i; \Omega) = P_{3d}^{cyl}(\mathbf{x}_i; \Omega), \quad \text{where } \mathbf{x}_i \in \mathcal{X}_s.$$

Finally in Case 3, both the assumed wing surface and wing contour models are incorrect. Again, since the wing contour is assumed to be the projection of a jump edge, no contour data is used in fitting. The fitting function is

$$F(\mathbf{x}_i; \Omega) = P_{3d}^{sph}(\mathbf{x}_i; \Omega), \quad \text{where } \mathbf{x}_i \in \mathcal{X}_s.$$



(a)

$$\text{Case 1: } F(\mathbf{x}_i; \Omega) = w_1(\mathbf{x}_i \in \mathcal{X}_s) P_{3d}^{cyl}(\mathbf{x}_i; \Omega) + w_2(\mathbf{x}_i \in \mathcal{X}_c) P_{2d}^{cyl}(\mathbf{x}_i; \Omega).$$

$$\text{Case 2: } F(\mathbf{x}_i; \Omega) = P_{3d}^{cyl}(\mathbf{x}_i; \Omega).$$

$$\text{Case 3: } F(\mathbf{x}_i; \Omega) = P_{3d}^{sph}(\mathbf{x}_i; \Omega).$$

(b)

Figure 3.6. Fitting wing models to a window with cylindrical surface patch. (a) Window from which the surface and edge points are sampled. (b) Fitting equations of three different wing models.

3.5 Experiments with Synthetic Data

Two sets of experiments with synthetic data were conducted to evaluate the performance of the fused fitting technique as compared to fitting surface or contour points alone.

In the first set of experiments, we set out to evaluate the quality of the fit on planar, spherical, cylindrical and conical surfaces when using respectively only contour data, only surface data and fused data. True surface models were used during each fit. The question to be answered by this set of experiments is *"Is the use of both contour and surface data for fitting superior to either used alone?"*

While the first set of experiments used true surface/contour models during fitting, the second set of experiments fitted both true and incorrect models to the data points. The goal is to determine if the wrong models will fit the surface/contour patch better than the true model. In other words, the experiments are to exploit the classification power of the fused data fitter against the usual techniques.

One thousand Monte Carlo trials were performed in each of the experiments. During each iteration, a new set of surface and contour data were synthetically generated as follows. First the parameters of the surface model were generated randomly. The ranges from which those parameters were generated are given in Table 3.1. Points on the surface could then be rendered without error using the selected parameter values and the appropriate surface model equation. A rectangular sampling window of specified size was placed over the image contour of the limb boundary. In the case of two intersecting planes, the window was placed over the line in the image that corresponded to the crease edge in 3D. Surface and contour points for fitting were sampled from this window. The total number of points sampled in each of the trials was 30. For surface points, Gaussian noise was added to the z -component to simulate sensor errors. Results at two noise levels, $\sigma = 0.01$ and $\sigma = 0.05$ are reported

Table 3.1. Range (in inches) of parameters

Planar:	$-5.0 \leq a_1, b_1, d_1, a_2, b_2, d_2 \leq 5.0.$		
Spherical:	$-5.0 \leq x_0, y_0 \leq 5.0;$	$15.0 \leq z_0 \leq 25.0;$	$1.0 \leq r_0 \leq 10.0.$
Cylindrical:	$-5.0 \leq x_0, y_0 \leq 5.0;$	$1.0 \leq r_0 \leq 10.0;$	$5.0 \leq \alpha, \beta \leq 175.$
Conical:	$-5.0 \leq x_0, y_0 \leq 5.0;$	$15.0 \leq z_0 \leq 25.0;$	$0.5 \leq d_0 \leq 2.0;$
	$5.0 \leq \alpha, \beta \leq 175.$		

here. Gaussian noise at levels of $\sigma = 0.002$ and $\sigma = 0.01$ was also added to both x - and y -components of the contour data points. In reality, the noise level of our range scanner [117] was determined to have $\sigma < 0.01$. We also experimented at a higher σ because we feel that range data derived using shape-from-x techniques would have a much higher noise level.

3.5.1 Fitting True Models to Contour, Surface, and Fused Data

The first four experiments involve fitting the true model to spherical, cylindrical, conical and planar surface patches. We compare the results of fitting fused (both contour and surface) data to fitting contour-only or surface-only data. The fused data consists of 15 points sampled from the contour and 15 points sampled from the adjacent surface patch for a total of 30 points. For surface-only and contour-only data sets, 30 points were sampled from either the surface patch or the contour in the window.

Since true model parameter values are known, the initial parameter estimates are taken to be the truth plus/minus some random deviations having uniform distribution. Table 3.2 lists the range of deviations for all of the parameters.

In studying the results of the experiments, one of the statistics reported is the total number of bad fits over the 1000 trials. A fit is considered bad if the computed χ^2 is over the threshold of 12. Since 30 data points were used in each trial, and the

Table 3.2. Initial estimate of the model parameters

Planar		Conical	
$\hat{a}_1 = a_1 \pm da_1$,	$da_1 \sim U(0, 1)$	$\hat{x}_0 = x_0 \pm dx_0$,	$dx_0 \sim U(0, 1)$
$\hat{b}_1 = b_1 \pm db_1$,	$db_1 \sim U(0, 1)$	$\hat{y}_0 = y_0 \pm dy_0$,	$dy_0 \sim U(0, 1)$
$\hat{d}_1 = d_1 \pm dd_1$,	$dd_1 \sim U(0, 1)$	$\hat{z}_0 = z_0 \pm dz_0$,	$dz_0 \sim U(0, 1)$
$\hat{a}_2 = a_2 \pm da_2$,	$da_2 \sim U(0, 1)$	$\hat{d}_0 = d_0 \pm dd_0$,	$dd_0 \sim U(0, 0.5)$
$\hat{b}_2 = b_2 \pm db_2$,	$db_2 \sim U(0, 1)$	$\hat{\alpha} = \alpha \pm d\alpha$,	$d\alpha \sim U(0, 10)$
$\hat{d}_2 = d_2 \pm dd_2$,	$dd_2 \sim U(0, 1)$	$\hat{\beta} = \beta \pm d\beta$,	$d\beta \sim U(0, 10)$
Spherical		Cylindrical	
$\hat{x}_0 = x_0 \pm dx_0$,	$dx_0 \sim U(0, 1)$	$\hat{x}_0 = x_0 \pm dx_0$,	$dx_0 \sim U(0, 1)$
$\hat{y}_0 = y_0 \pm dy_0$,	$dy_0 \sim U(0, 1)$	$\hat{y}_0 = y_0 \pm dy_0$,	$dy_0 \sim U(0, 1)$
$\hat{z}_0 = z_0 \pm dz_0$,	$dz_0 \sim U(0, 1)$	$\hat{r}_0 = r_0 \pm dr_0$,	$dr_0 \sim U(0, 2)$
$\hat{r}_0 = r_0 \pm dr_0$,	$dr_0 \sim U(0, 1)$	$\hat{\alpha} = \alpha \pm d\alpha$,	$d\alpha \sim U(0, 10)$
		$\hat{\beta} = \beta \pm d\beta$,	$d\beta \sim U(0, 10)$

number of parameters is either 4 (spherical), 5 (cylindrical) or 6 (conical, planar), the degrees of freedom would then be 26, 25 or 24. A χ^2 of 12 with number of degrees of freedom in the range of 24 - 26 indicates that the probability of the fit obtaining a χ^2 as small as 12 happening by chance is about 2%. While a bad fit is a type II error in the hypothesis testing sense in which the correct hypothesis is falsely rejected, we note that additional reject conditions will be added in the following subsections, making the term "type II error" an inaccurate description of the rejects by the whole model fitting procedure. Thus, we shall use the term "bad fits" to denote those fits that resulted in a type II error or rejected by the additional "good" fit criteria.

All other statistics are computed over the good fits only. Besides reporting the average and maximum deviations of the estimated parameters from the ground truth, three other summarizing statistics are also reported. They are the average *d.lse*, the average *m.lse* and the average number of iterations for the good fits to converge. The *d.lse* models the variance of the Gaussian noise in the data and is computed as the unweighted sum of the squares of the Euclidean distances between the data points and the fitted surface/contour averaged over all 30 points. The *m.lse* is a similar

Table 3.3. Fitting Spherical Model on Spherical Data

Number of trials: 1000							
Number of samples per trial: 30							
Parameter Measurements are in inches		Contour-Only		Surface-Only		Fused	
		Avg.	Max.	Avg.	Max.	Avg.	Max.
$\sigma = 0.01$	$\hat{x}_0 - x_{true}$	0.00	0.01	0.01	0.03	0.00	0.01
	$\hat{y}_0 - y_{true}$	0.04	0.45	0.03	0.14	0.02	0.08
	$\hat{z}_0 - z_{true}$	Inf.	Inf.	0.03	0.14	0.01	0.04
	$\hat{r}_0 - r_{true}$	0.04	0.50	0.04	0.19	0.02	0.08
	χ^2	0.01658		0.26373		0.13292	
	$d\ lse$	0.00001		0.00009		0.00004	
	$m\ lse$	0.00001		0.00002		0.00001	
	<i>iterations</i>	3.99		4.29		5.37	
	<i>bad fits</i>	30		0		3	
$\sigma = 0.05$	$\hat{x}_0 - x_{true}$	0.01	0.03	0.03	0.11	0.01	0.06
	$\hat{y}_0 - y_{true}$	0.12	0.45	0.17	0.38	0.08	0.35
	$\hat{z}_0 - z_{true}$	Inf.	Inf.	0.18	0.43	0.04	0.17
	$\hat{r}_0 - r_{true}$	0.12	0.45	0.23	0.50	0.08	0.34
	χ^2	0.27139		6.80193		3.19649	
	$d\ lse$	0.00009		0.00227		0.00107	
	$m\ lse$	0.00001		0.00056		0.00015	
	<i>iterations</i>	4.12		4.72		5.77	
	<i>bad fits</i>	34		188		3	

quantity except the error is computed from the data points *before Gaussian noise was added* to the fitted surface/contour. This measure gives an indication of how well the estimated surface/contour matches the noise free data.

Experiment 1: Spherical

The sampling window size was chosen to be proportional to the radius of the sphere as $0.5r_0 \times 0.6r_0$. This covers roughly 10% of the projected area of the hemisphere. Since the sampled area is a small fraction of the sphere, many good fits may result with various different radii. An additional "good fit" criterion is added to detect those fits whose estimated radius deviates by more than 0.5 inches from the ground

truth. Thus, a fit is said to be good if and only if $\chi^2 < 12.0$ and the estimated radius is no more than 0.5 inches from the truth. The results are summarized in Table 3.3.

In fitting contour-only data, the z -component of the center of the sphere cannot be recovered. When the noise contamination level is low ($\sigma = 0.01$), the results are comparable between surface-only and fused fitting. Perhaps a slight advantage goes to fused fitting. In fitting with fused data, the estimated parameters are closer to the ground truth. The average number of iterations is only one more than if only surface data were used. Although there are 3 bad fits using fused data, it is negligible in light of the large number of trials. The results in the more noisy case ($\sigma = 0.05$) strongly support our claim that fused fitting is superior then fitting with 3D or 2D data alone. The number of bad fits using fused data remained low while the number of bad fits for surface-only data shot up to 18.8%. The estimated radii are also much worse using surface data alone.

Overall, the results suggest that simultaneously fitting surface and contour points can tolerate a much higher noise level. The fitted parameters are more accurate, and these results are obtained without much increase in computation time (one more iteration).

Experiment 2: Cylindrical

The window from which the data points are to be sampled is a rectangular box of height $2.0r_0$ (running parallel to the axis of the cylinder) and width $0.6r_0$. This roughly covers 33% of the projected area of a cylinder of height $2.0r_0$. As in the spherical case, many fits will be deemed good if χ^2 were the only criterion used. Here a fit is consider good if and only if (1) $\chi^2 < 12.0$, (2) $|\hat{r}_0 - r_0| \leq 0.5$, and (3) $|\hat{\alpha} - \alpha| \leq 10^\circ$ and $|\hat{\beta} - \beta| \leq 10^\circ$. Those conditions force a good fitted cylinder to have similar size and have the axis approximately pointing in the true direction. Results from the Monte Carlo trials are depicted in Table 3.4.



Table 3.4. Fitting Cylindrical Model on Cylindrical Data

Number of trials: 1000							
Number of samples per trial: 30							
Parameter Measurements are in inches		Contour-Only		Surface-Only		Fused	
		Avg.	Max.	Avg.	Max.	Avg.	Max.
$\sigma = 0.01$	$\hat{x}_0 - x_{true}$	1.69	61.84	0.29	14.52	0.12	2.81
	$\hat{y}_0 - y_{true}$	0.44	5.33	0.17	5.27	0.08	1.92
	$\hat{r}_0 - r_{true}$	0.25	0.50	0.09	0.46	0.03	0.27
	$\hat{\alpha} - \alpha_{true}$	4.59	9.98	0.63	7.52	0.60	5.19
	$\hat{\beta} - \beta_{true}$	2.70	9.71	0.49	6.88	0.48	4.30
	χ^2	0.06889		0.26924		0.18603	
	$d\ lse$	0.00002		0.00009		0.00006	
	$m\ lse$	0.00002		0.00725		0.00067	
	<i>iterations</i>	3.45		7.18		6.63	
	<i>bad fits</i>	815		58		77	
$\sigma = 0.05$	$\hat{x}_0 - x_{true}$	1.40	18.74	0.54	16.39	0.50	14.51
	$\hat{y}_0 - y_{true}$	0.43	5.44	0.29	3.29	0.35	10.86
	$\hat{r}_0 - r_{true}$	0.25	0.50	0.23	0.50	0.15	0.50
	$\hat{\alpha} - \alpha_{true}$	4.63	9.99	0.91	9.95	1.29	9.89
	$\hat{\beta} - \beta_{true}$	2.61	9.70	0.62	5.02	0.70	5.37
	χ^2	0.32513		6.63258		4.31621	
	$d\ lse$	0.00011		0.00221		0.00144	
	$m\ lse$	0.00003		0.00087		0.00369	
	<i>iterations</i>	3.72		4.76		6.18	
	<i>bad fits</i>	819		676		186	



As with the spherical cases, the fused fit out-performed the other two methods. This is especially true with the noisier data. Note that the contour-only based fitting converged on less than 20% of all trials performed and is therefore not a good choice for parameter estimations. Under light noise contamination, the convergence rate for the surface-only fit is 2% higher than the fused fit. However, the fused fits result in better parameter estimates and slightly faster rate of convergence. With higher noise level, the surface-only fits failed much more frequently and the estimated parameters were inferior to those obtained in fused fits.

Experiment 3: Conical

The size and placement of the window is chosen as follows. First a point x_w along the limb boundary at some random elevation from the vertex of the cone is selected. The radius of the circular cross-section that touches x_w and orthogonal to the axis can be then be computed, say r_0 . The window from which the data points are to be sampled is thus a rectangle of height $2.0 r_0$ (running parallel to the limb contour of the cone) and width $0.6 r_0$ placed with one corner of the window coinciding with the selected point on the limb contour. The good fit criteria are similar to the cylindrical case, which are: (1) $\chi^2 < 12.0$, (2) $|\hat{d}_0 - d_0| \leq 0.5$, and (3) $|\hat{\alpha} - \alpha| \leq 10^\circ$ and $|\hat{\beta} - \beta| \leq 10^\circ$. Results of this experimentation are given in Table 3.5.

As with the previous two cases, the contour-only fits are not reliable for 3D parameter estimation. In low noise, the fused fits performed the best in all category including number of bad fits, the accuracy of the estimated parameters and the convergence rate. However, none of the fitting methods performed satisfactorily when the noise level was raised to $\sigma = 0.05$. Overall, conical model fitting performed worse than the spherical and cylindrical cases, probably because the conical model has more degrees of freedom and the region from which the data points are sampled is smaller [24].

Table 3.5. Fitting Conical Model on Conical Data

Number of trials: 1000							
Number of samples per trial: 30							
Parameter Measurements are in inches		Contour-Only		Surface-Only		Fused	
		Avg.	Max.	Avg.	Max.	Avg.	Max.
$\sigma = 0.01$	$\hat{x}_0 - x_{true}$	0.41	1.00	0.33	2.99	0.27	2.96
	$\hat{y}_0 - y_{true}$	0.38	0.97	0.28	1.87	0.21	1.69
	$\hat{z}_0 - z_{true}$	0.51	1.00	0.55	11.73	0.37	8.98
	$\hat{d}_0 - d_{true}$	0.20	0.47	0.09	0.49	0.08	0.46
	$\hat{\alpha} - \alpha_{true}$	4.33	9.93	2.31	9.99	1.91	9.96
	$\hat{\beta} - \beta_{true}$	4.13	9.67	1.31	9.45	1.12	9.78
	χ^2	4.07391		0.10676		1.12798	
	$d\ lse$	0.00136		0.00004		0.00038	
	$m\ lse$	0.00136		0.00921		0.01186	
	<i>iterations</i>	5.15		15.03		12.53	
	<i>bad fits</i>	843		273		155	
$\sigma = 0.01$	$\hat{x}_0 - x_{true}$	0.40	1.00	0.55	3.22	0.48	2.92
	$\hat{y}_0 - y_{true}$	0.37	0.97	1.02	3.92	0.51	2.25
	$\hat{z}_0 - z_{true}$	0.51	1.00	2.33	15.65	0.91	11.08
	$\hat{d}_0 - d_{true}$	0.20	0.47	0.17	0.48	0.11	0.49
	$\hat{\alpha} - \alpha_{true}$	4.17	9.93	3.87	9.71	3.14	9.86
	$\hat{\beta} - \beta_{true}$	3.96	9.59	2.55	8.95	2.10	9.64
	χ^2	3.85353		1.70781		5.00942	
	$d\ lse$	0.00128		0.00057		0.00167	
	$m\ lse$	0.00121		0.04439		0.00210	
	<i>iterations</i>	5.14		27.44		17.45	
	<i>bad fits</i>	854		872		744	

Table 3.6. Fitting Planar Model on Planar Data

Number of trials: 1000					
Number of samples per trial: 30					
Parameter Measurements are in inches		Surface-Only		Fused	
		Avg.	Max.	Avg.	Max.
$\sigma = 0.01$	$\hat{a}_1 - a_{true}$	0.00	0.02	0.00	0.02
	$\hat{b}_1 - b_{true}$	0.00	0.02	0.00	0.02
	$\hat{d}_1 - d_{true}$	0.01	0.73	0.01	0.14
	$\hat{a}_2 - a_{true}$	0.00	0.02	0.00	0.02
	$\hat{b}_2 - b_{true}$	0.00	0.02	0.00	0.02
	$\hat{d}_2 - d_{true}$	0.01	0.62	0.01	0.36
	χ^2	0.43084		0.44556	
	$d\ lse$	0.00014		0.00015	
	$m\ lse$	0.00001		0.00001	
	iterations	3.53		3.39	
	bad fits	0		0	
$\sigma = 0.05$	$\hat{a}_1 - a_{true}$	0.00	0.02	0.00	0.02
	$\hat{b}_1 - b_{true}$	0.00	0.02	0.00	0.02
	$\hat{d}_1 - d_{true}$	0.01	0.55	0.01	0.29
	$\hat{a}_2 - a_{true}$	0.00	0.02	0.00	0.02
	$\hat{b}_2 - b_{true}$	0.00	0.02	0.00	0.02
	$\hat{d}_2 - d_{true}$	0.01	0.29	0.01	0.32
	χ^2	6.58697		6.58583	
	$d\ lse$	0.00220		0.00220	
	$m\ lse$	0.00007		0.00004	
	iterations	3.85		3.64	
	bad fits	13		13	

Experiment 4: Planar

The fused fit is performed over two adjacent planar surfaces along with the projection of the intersection edge. No contour-only fitting is performed because there is insufficient constraint on the 3D parameters. The sampling window size is fixed at 1.0×2.0 , with each surface getting 1.0×1.0 of coverage. The surface-only data set consisted of 15 points from each of the two surfaces whereas the fused data set is made up of 10 surface points from each surface and 10 contour points. A good fit is simply a fit in which $\chi^2 < 12$. The results are shown in Table 3.6.

The differences in the planar fits are judged to be insignificant. Almost all 2000 Monte Carlo trials converged to the ground truth. The fused fit is slightly faster than the surface-only fit. However, the difference is small. One observation that came out of additional Monte Carlo trials using a wide range of initial parameter values is that planar fitting is very tolerant of bad initial guesses. The fits still converge rapidly even when the starting parameter values are far off the final values. This is consistent with many reports of planar fitting in the literature. All of this implies that fused fitting is not needed for polyhedra.

3.5.2 Surface Classification Experiments

The next 4 experiments are designed to test the surface classification and shape recognition power of fitting fused data as compared to fitting surface-only data. Specifically, we want to test how well the fused fits reject the fitting of one model to data generated from other models (*e.g.*, fitting all models to spherical surface path).

The initial guesses of the parameter values are important in finding the correct fit. When the true model is fitted to the surface patch, the initial parameter estimates are perturbed from the truth as before. However, when an incorrect model is fitted, the initial estimate of the parameters can no longer be directly generated from the true model parameters of data to be fitted. Care has been taken to ensure that the initial guess results in a surface close to the actual surface patches. The reported statistics include the number of bad fits, the average χ^2 , the average number of iterations and 2 new entities. First, *Best χ^2* counts the frequency in which one model is the best fit among all good fits in terms of χ^2 . If all models failed to fit a surface patch, then the comparison is not made. The other new entry, *misclassification %*, is the percentage in which the wrong surface model fitted the surface patch better than the true surface

Table 3.7. Fitting all models to Spherical surface patch

True surface form: SPHERICAL									
Number of trials: 1000									
Number of points sampled per trial: 30									
		Surface-Only				Fused			
		con	cyl	sph	pln	con	cyl	sph	pln
σ 0.01	<i>bad fits</i>	909	996	0	931	1000	1000	0	855
	χ^2	8.77	10.81	0.27	7.62	-	-	0.13	6.15
	<i>iterations</i>	19.25	14.25	3.78	3.62	-	-	4.79	3.31
	<i>Best χ^2</i>	0	0	1000	0	0	0	1000	0
	% misclassified	0.0%				0.0%			
σ 0.05	<i>bad fits</i>	939	1000	99	972	1000	1000	0	886
	χ^2	9.19	-	6.85	9.49	-	-	3.34	7.08
	<i>iterations</i>	22.08	-	4.33	3.5	-	-	5.44	3.39
	<i>Best χ^2</i>	22	0	888	14	0	0	998	2
	% misclassified	3.9%				0.2%			

model. Again, if all surface models failed to converge, then no classification was performed and that trial was not included in the calculation of frequency of misclassification.

Experiment 5: Fitting all models to spherical patch

The results of fitting all models to a spherical surface patch are depicted in Table 3.7. When the added Gaussian noise is low ($\sigma = 0.01$), there is no misclassification error using either data set. Note that when contour information is not used, the conical model will fit the spherical surface patch 9% of the time, whereas when contour information was used, the conical model never converged. The effect of added contour information is more apparent in the more noisy case. Using fused data, the misclassification rate was 0.2%. Using surface-only data, the error rate jumps to 3.9%. Again, when contour data was also used during fitting, the conical and cylindrical models never fitted well. Since the limb contour of a sphere is curved while the limb contour of a cylinder or cone is straight, the use of contour points will cause the fitter to

reject cylindrical and conical models as bad fits. This further supports our claim that contour information is important in classifying surface shape. We must make a note that when fitting a planar model to the surface patch, only the surface data is used.

Observation: Among the wrong models, the conical model fit the spherical surface patch the best only when surface data was used. With fused data, planar model does better among the wrong models.

Experiment 6: Fitting all models to Cylindrical patch

The cylindrical surface/contour data sets were generated as in Experiment 2 except for the height of the window (along the limb boundary) which was shrunk from $2.0r_0$ down to $0.5r_0$. With the original elongated window, the spherical and conical models would be quickly ruled out by the long straight limb and zero curvature along the limb. With a more squared sampling window ($0.5r_0 \times 0.6r_0$), The conical and spherical models have a better chance of fitting the cylindrical surface patch and the contribution of the contour data in fused fitting can be better interpreted. Results of the experiments are given in Table 3.8.

With low Gaussian noise, the cylindrical model has the best fit most of the time. Although the misclassification rate for fused fits (4.5%) is lower than that of surface-only fits (7.8%), they are close. With higher noise level, the power of shape recognition with fused data is more apparent. While fitting surface-only data yields a 60.8% misclassification rate, fused data fitting has a much lower 16.6% error rate. Furthermore, the conical model fitted the cylindrical surface patch more successfully than the cylindrical model when only surface points were used, but this tendency was reversed with fused data. This can be attributed to the fact that the Gaussian noise induced surface patch no longer resembles a cylindrical patch. Since a conical model has one

Table 3.8. Fitting all models to Cylindrical surface patch

True surface form: CYLINDRICAL									
Number of trials: 1000									
Number of points sampled per trial: 30									
		Surface-Only				Fused			
		con	cyl	sph	pln	con	cyl	sph	pln
σ 0.01	<i>bad fits</i>	530	58	882	736	824	61	913	571
	χ^2	5.38	0.30	6.76	5.63	5.11	0.21	6.10	5.21
	<i>iterations</i>	23.72	6.80	8.53	3.61	29.00	6.82	4.55	3.41
	<i>Best χ^2</i>	58	905	6	13	4	933	0	40
	% misclassified	7.8%				4.5%			
σ 0.05	<i>bad fits</i>	661	728	996	857	859	233	942	637
	χ^2	7.17	6.53	10.09	8.85	6.09	4.39	7.94	6.87
	<i>iterations</i>	21.16	5.43	4.00	3.63	29.10	6.51	4.67	3.45
	<i>Best χ^2</i>	276	235	1	87	45	706	0	96
	% misclassified	60.8%				16.6%			

more degree of freedom, it would fit this "irregular" surface patch better than the cylindrical model. When the limb contour is introduced in fused fitting, it forces the limb of the conical model to line up with the contour data and consequently makes the tapering surface of the conical model a bad fit to the non-tapering surface patch. On the other hand, the contour data put a fix on the axis direction of the cylindrical model which wasn't available to surface-only fitting. Consequently, it would fit better than before.

In fitting surface-only data set, when the surface is misclassified, the conical model tends to have the best fit, followed by planar then spherical models. We must note that although a large sphere may fit a small cylindrical patch well, we have made an explicit assumption that no sphere of radius greater than 10 inches is possible (see Table 3.1). Thus, all spherical fits with $r_0 > 10$ were rejected.

On the other hand, if the surface patch is misclassified when fitting fused data, the planar model tends to have the best fit followed by the conical model but never the spherical model.

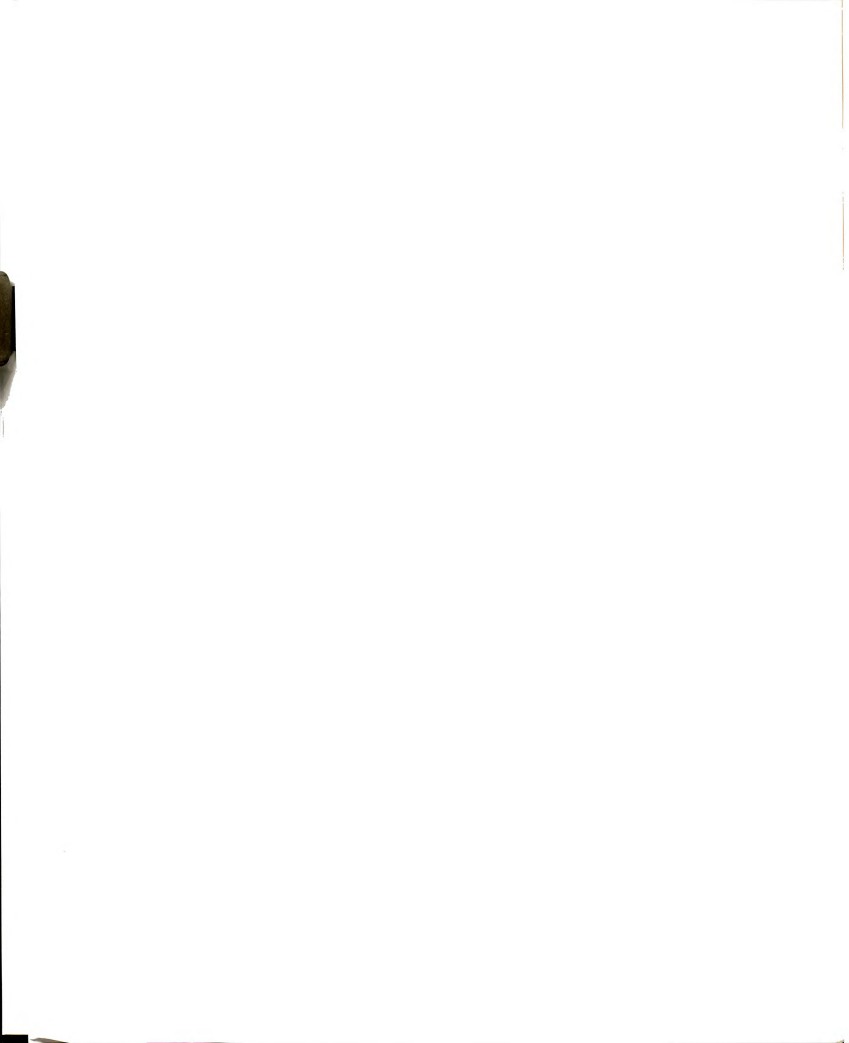


Table 3.9. Fitting all models to Conical surface patch

True surface form: CONICAL									
Number of trials: 1000									
Number of data points sampled per trial: 30									
		Surface-Only				Fused			
		con	cyl	sph	pln	con	cyl	sph	pln
σ 0.01	<i>bad fits</i>	153	775	745	1000	62	935	998	1000
	χ^2	0.15	3.62	2.67	-	1.17	4.74	9.09	-
	<i>iterations</i>	15.78	21.53	6.55	-	12.86	15.74	7.00	-
	<i>Best χ^2</i>	835	20	129	0	934	24	0	0
	% misclassified	15.1%				2.5%			
σ 0.05	<i>bad fits</i>	784	859	803	1000	642	949	1000	1000
	χ^2	2.18	8.13	7.43	-	5.30	7.08	-	-
	<i>iterations</i>	30.51	20.03	7.05	-	18.51	16.53	-	-
	<i>Best χ^2</i>	215	98	183	0	357	49	0	0
	% misclassified	56.7%				13.7%			

Observation: Among the wrong models, the conical model fit the cylindrical surface patch the best if only surface data were used. With fused data, the the planar model does better among the wrong models.

Experiment 7: Fitting all models to Conical patch

The result of this experiment is consistent with the previous experiments in that fitting with fused data yielded fewer misclassifications. From Experiment 3, we learned that fitting a small conical surface patch is a difficult task even if a conical model were given. Here we found out that a wrong model is less likely to fit a small conical surface patch if fused data were used. From Table 3.9, the conical surface patch is misclassified 15.1% ($\sigma = 0.01$) and 56.7% ($\sigma = 0.05$) of the time when only surface points are fitted. When both surface and limb contour points are used, the misclassification rate drops to 2.5% and 13.7%. We also note that without contour information, among the misclassified cases, the spherical model tends to fit the surface patch well, while the spherical model was never a good fit to the conical surface patch when fused data

were used. This phenomenon is also consistent with the earlier observation that the straightness of the limb contour prohibits the spherical model from being a good fit to the conical surface patch.

Observation: Among the wrong models, the spherical model fit the conical surface patch the best if only surface data were used. With fused data, the cylindrical model does better among the wrong models.

Experiment 8: Fitting all models to Planar patch

As in Experiment 4, data points were sampled from a window of size $1.00 \times 2.00 \text{ in}^2$. The planar model was fitted to the data set the same way as before. However, when fitting conical, cylindrical and spherical models, only data collected from one of the planar surfaces was used. Any small planar patch can be reasonably fitted if the radius of the quadric model is large. For this reason, the initial r_0 was set to 7 inches for cylindrical and spherical models. The other parameters were chosen carefully so that the surface would conform close to the planar patch. As in the previous two experiments, any cylindrical or spherical fit with radius greater than 10 inches was rejected as a bad fit. The results of this experiment are given in Table 3.10.

Clearly, fitting with fused data results in much better classification power. The fused fit only misclassified 1 surface patch out of 2000 chances where the surface-only fit produces a much worse result. As with previous observations, without the edge contour data, the conical model will fit the surface patch rather well. However, with the added contour information, only one of the wrong models fitted the planar patch better than the planar model.

Table 3.10. Fitting all models to Planar surface patch

True surface form: PLANAR									
Number of trials: 1000									
Number of data points sampled per trial: 30									
		Surface-Only				Fused			
		con	cyl	sph	pln	con	cyl	sph	pln
σ 0.01	<i>bad fits</i>	178	937	998	0	1000	996	1000	0
	χ^2	3.54	3.63	6.91	0.42	-	8.22	-	0.41
	<i>iterations</i>	15.33	31.43	49	3.47	-	28.25	-	3.30
	<i>Best χ^2</i>	161	9	0	830	0	0	0	1000
	<i>% misclassified</i>	17.0%				0.0%			
σ 0.05	<i>bad fits</i>	277	974	998	0	1000	996	1000	0
	χ^2	4.66	5.69	7.25	6.48	-	8.79	-	6.47
	<i>iterations</i>	17.76	37.77	42.5	3.84	-	23.00	-	3.59
	<i>Best χ^2</i>	483	7	0	510	0	1	0	999
	<i>% misclassified</i>	49.0%				0.1%			

Observation: Among the wrong models, the conical model fit the planar surface patch the best only when surface data were used. With fused data, none of the wrong models fit the planar patch well.

3.5.3 Simulation Results Summary

The simulation results of the first 4 experiments show that the use of fused data in surface parameter recovery is superior to using either the contour-only or the surface-only data. In general, the fused fit yields more accurate parameter estimates and better convergence rate (especially under higher noise level) without a significant increase in number of iterations before convergence. In fact, the average number of iterations for convergence went down using the fused data in some cases.

Monte Carlo experiments directed toward recognition capability indicate that the proposed fused fitting method has a much lower false alarm rate than the method of fitting only surface data. On the rare occasions where the fit to an incorrect model



Table 3.11. Percentage of bad fused fits versus number of surface and contour points.

n \ m	5	10	15	20	25
5	32.7%	8.0%	7.6%	5.9%	5.2%
10	31.8%	7.8%	7.7%	6.2%	5.6%
15	31.3%	7.9%	7.1%	6.5%	5.9%
25	30.6%	7.8%	7.6%	6.4%	5.7%
m = number of surface points					
n = number of contour points					

does converge, the misclassification rate (wrong surface label) is also much lower. This misclassification rate appears to be small enough to give good input to our planned higher level scene reconstruction module. The observations made in each of the experiments strongly supports our hypothesis that edge contour information contributes greatly to the correct qualitative classification of surface shape.

3.6 Fitting with Different Number of Surface and Contour Points

In the literature, fitting is often performed with a large number of surface or contour points. We argue that more is not necessarily better. In this section, we study the effect of fused and surface-only data fitting with various numbers of data points over a small cylindrical surface patch.

A Monte-Carlo experiment of 1000 trials with different numbers of surface and contour points was conducted. The number of surface points sampled was 5, 10, 15, 25, or 50 and the number of contour points was 5, 10, 15 or 20. The numbers of bad fits over the 1000 trials are reported in Tables 3.11. Contrary to popular belief and common practice in fitting, the results suggest that the performance of the fitter is stabilized with very few surface and contour points. Improving the performance by



Table 3.12. Surface-only fitting with various numbers of surface points.

m	Bad Fit %	m	Bad Fit %
10	38.6%	40	5.0%
15	47.4%	50	4.9%
20	6.1%	55	4.5%
25	5.1%	60	4.5%
30	5.1%	65	4.2%
35	5.6%	75	4.3%

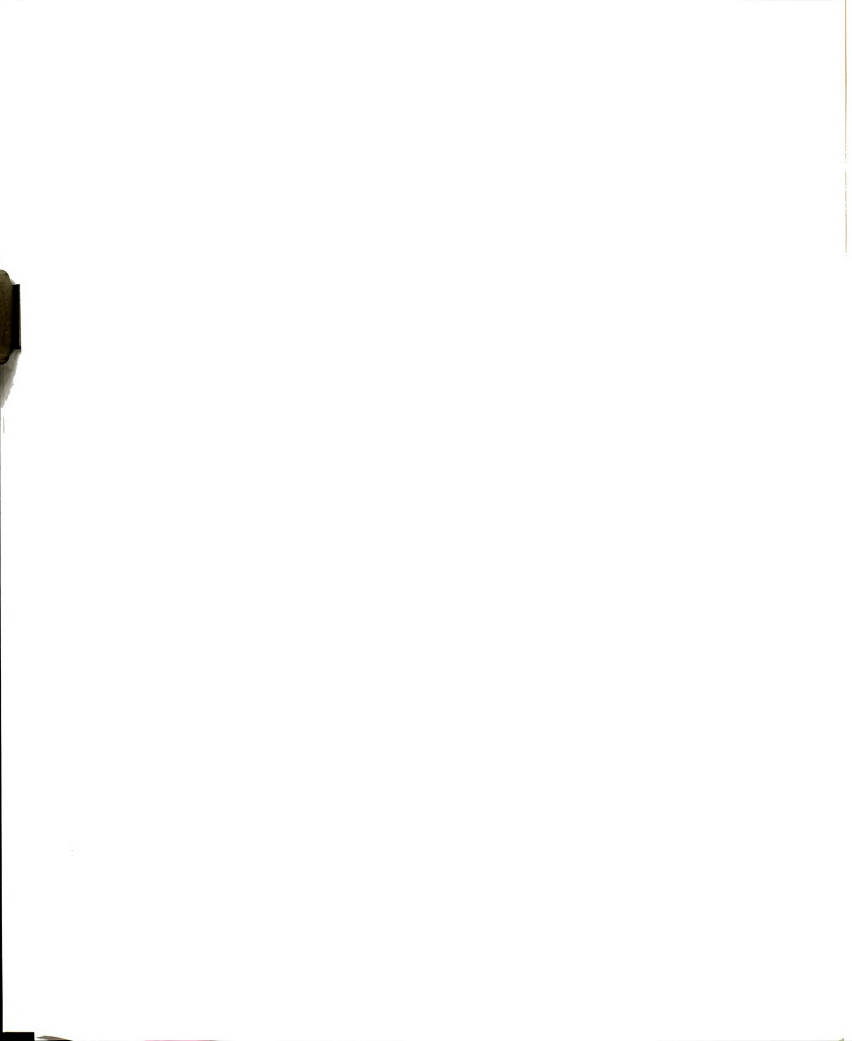
increasing the number of data points is possible but the gain is minimal compared to the increase of the data size.

Similar experiments on surface-only fitting was also conducted and reported in Table 3.12. The total number of points sampled ranges from 10 to 75. As with the fused fitting, the performance of the fit stabilized at a very low number of data points (20). Although the data suggests that more data points would improve the fit, the improvement is small. One must weigh the advantage of higher convergence ratio to the cost of slower convergence rate and more expensive sensing when more points are used to determine the number of points to use for fitting. Similar results should be expected with surfaces other than cylindrical surfaces.

3.7 Experiments with Real Fused Imagery

3.7.1 JIG1 Example

All surfaces in this image are planar (see Figure A.4). 104 out of total 255 windows were interesting windows, (*i.e.*, candidate windows for wing detection). Of the 104 interesting windows, 80 were found to contain wings. All hypotheses in the other 24 windows were rejected due to poor surface/fused fits. Of the 80 windows, 75 wings were correctly identified. However, 5 false wings were also reported. Those 5 false wings all belong to the error category of *E2*. *E2* occurs if the window covers an

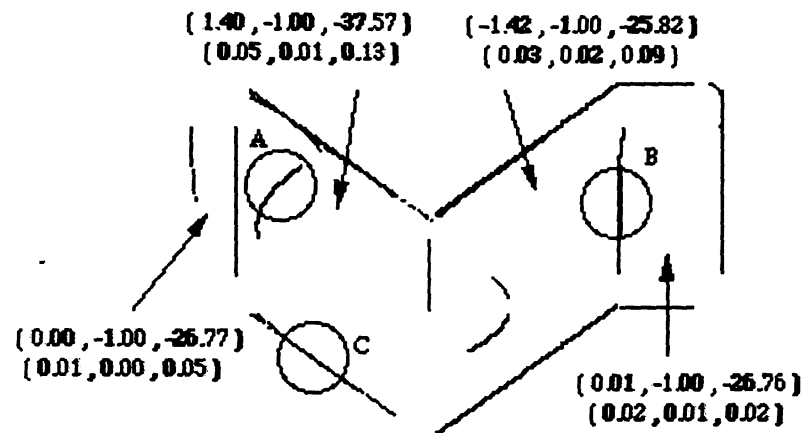


homogeneous region of the surface but the wing detector managed to find a wing in it. For example, in Figure 3.7, window A produces a spurious wing. Window B shows that the correct wing primitive hypothesis, $(PLN+LN+PLN)$, is verified as the wing present in the window and the other false hypotheses are rejected by fused fitting. Window C shows that a jump wing is correctly detected.

To evaluate the fitted parameters, we manually clustered the 75 good wings into the 4 surfaces (this step would be done automatically by the LDG reconstruction process) and visually inspected the surface parameters of wings in the same cluster. The means and the standard deviations of each parameter are also shown in Figure 3.7 (The three numbers represent (A, B, D) in the planar equation $Ax + By + z + D = 0$). From the standard deviations, we see that there is little variance in parameters among all wings in the same cluster. Note that the two surfaces at the far ends have the same normals and the dot product of the normals of the two inner surfaces, which are perpendicular to each other, equals 0.

3.7.2 CYL2 Example

There are two types of surfaces in this image: cylindrical and planar (see Figure A.4). One of the limbs is also registered in the fused imagery. Results of the wing detection indicates that there are only 45 interesting windows: 30 winged windows and 15 busy windows. Of the 30 winged windows, 23 contain *good* wings and 7 contain erroneous wings. The errors are of type *E1* (4) and type *E3* (3). An *E1* error is one where the simple wing test failed to recognize that two or more 3D edges exist in the window and to classify it as a busy window (*e.g.*, window F and the lower left corner of Figure 3.8). An *E3* error is one where the surface is cylindrical but the cylindrical surface hypothesis was rejected and the planar/spherical hypotheses was accepted. All 3 *E3* errors had occurred in windows in which only surface data were available for fitting (*i.e.*, along the boundary near the lower limb in Figure 3.8).



```

A:(90 30) wing! - PLN, PLN
  hypotheses  chi_sq fitted parameters
  (CYL+PB+    1.44) (CYL: x0=92.51 y0= 7.47 r0= 43.43 alpha= 31 beta=136)
  *(PLN+PB+    1.33) (PLN: a = 1.47 b =-0.99 d =-38.01)
  (SPH+PB+    7.79) (SPH: x0=13.87 y0=-4.93 z0= 32.86 r0= 9.92)
  (  +PB+CYL 159.30) (CYL: x0=61.36 y0=25.63 r0= 27.97 alpha= 28 beta=148)
  *(  +PB+PLN  1.12) (PLN: a = 1.33 b =-1.04 d =-36.99)
  (  +PB+SPH 27.06) (SPH: x0= 8.18 y0=-0.99 z0= 28.59 r0= 1.88)
  ...

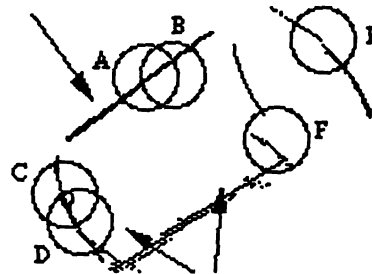
B:(90 165) wing! - PLN+LN+PLN
  (CYL+LN+PLN 1000) (CYL: x0= 0.00 y0= 0.00 r0= 0.00 alpha= 0 beta= 0)
  (PLN: a = 0.00 b = 0.00 d = 0.00)
  (PLN+LN+CYL 1000) (PLN: a = 0.00 b = 0.00 d = 0.00)
  (CYL: x0= 0.00 y0= 0.00 r0= 0.00 alpha= 0 beta= 0)
  *(PLN+LN+PLN 1.46) (PLN: a = 0.01 b = -1.00 d = -26.78)
  (PLN: a = -1.46 b = -1.00 d = -25.79)
  ...

C:(150 45) wing! - PLN
  (CYL+  + 1000) (CYL: x0=-43.49 y0= 23.04 r0=61.36 alpha= 50 beta=136)
  (CYL+LN+ 597) (CYL: x0= 37.28 y0=-16.43 r0= 0.00 alpha=153 beta= 46)
  (SPH+LN+ 1000) (SPH: x0= 3.55 y0= -0.87 z0=24.81 r0= 3.54)
  *(PLN+LN+ 1.06) (PLN: a = 1.39 b = -0.97 d =-37.43)
  ...

```

Figure 3.7. Detected wings in the jig image. Window A in the image corresponds to the output marked (90 30) where the jump wing hypothesis (PLN+LN), (LN+PLN) is wrongly accepted. It should be a bland window. Window B in the image corresponds to the output marked (90 165) where the crease wing hypothesis (PLN+LN+PLN) is correctly accepted and the others rejected. Window C in the image corresponds to the output marked (150 45) where the jump wing is detected. * denotes accepted hypothesis.

(-42.35, 33.77, 1.79, 51, 46)
 (0.83, 1.30, 0.06, 0.94, 0.00)

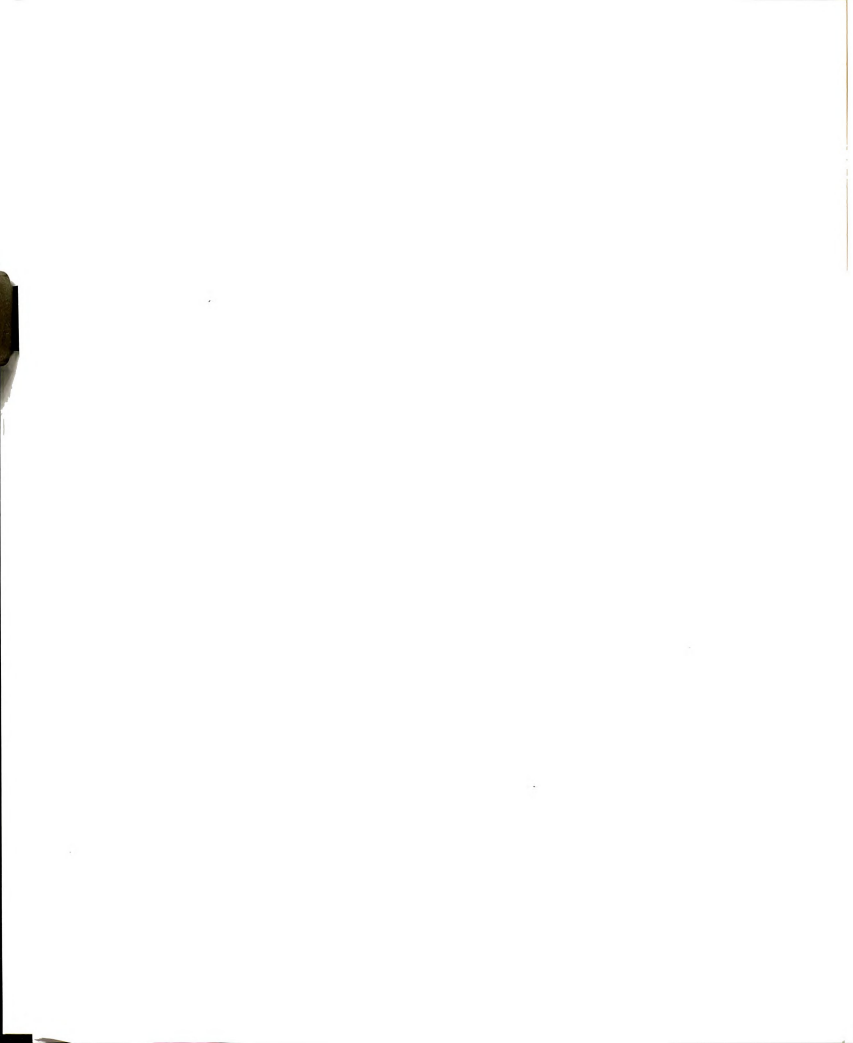


(-35.21, 28.47, 1.77, 47, 43)
 (5.33, 3.26, 0.31, 3.44, 4.71)

A: (60 105) wing! - CYL, CYL+LN
 *(+ +CYL 1.92) (CYL: x0=-37.67 y0=28.21 r0= 1.65 alpha= 46 beta= 45)
 *(+LN+CYL 3.42) (CYL: x0=-42.92 y0=34.84 r0= 1.74 alpha= 52 beta= 45)
 B: (60 120) wing! - CYL+LN
 (+ +CYL 10.44) (CYL: x0=-69.36 y0= 68.85 r0= 1.78 alpha= 68 beta= 43)
 *(+LN+CYL 1.78) (CYL: x0=-41.18 y0= 31.93 r0= 1.87 alpha= 50 beta= 45)
 C: (105 75) wing! - CYL, SPH
 *(CYL+ + 1.31) (CYL: x0=-32.56 y0= 23.17 r0= 1.83 alpha= 41 beta= 44)
 (CYL+LN+ 21.17) (CYL: x0=-20.19 y0=-78.03 r0= 0.87 alpha=109 beta=163)
 *(SPH+LN+ 1.34) (SPH: x0= 3.24 y0= -1.03 z0= 27.76 r0= 1.88)
 D: (120 90) wing! - SPH, CYL
 *(CYL+PB+ 5.39) (CYL: x0=-26.02 y0= 32.35 r0= 1.29 alpha= 50 beta= 34)
 *(SPH+PB+ 4.28) (SPH: x0= 3.29 y0= -0.94 z0= 27.37 r0= 1.56)
 E: (60 195) wing! - PLN
 *(PLN+CC+ 1.21) (PLN: a = 1.38 b = -1.05 d = -25.56)
 F: (90 165) wing! - (PLN, SPH)
 *(PLN+PB+ 0.42) (PLN: a = 1.61 b = -1.13 d = -25.54)
 *(+PB+SPH 0.57) (SPH: x0= 0.75 y0= 0.64 z0= 26.21 r0= 1.88)
 (+PB+CYL 1000) (CYL: x0= 0.00 y0= 0.00 r0= 0.00 alpha= 0 beta= 0)

...

Figure 3.8. Detected wings in the CYL image. Wings A and B : limb location in 2D aids the recovery of the cylinder parameters. Wings C and D : cylinder parameters recovered without the limb information in 2D. Wing E: correct wing segment type is recovered. Wing F: small cylindrical surface patch is best fitted by a spherical model. * denotes accepted hypothesis.



Again, we clustered the *correct* wings into the two surface regions. We further clustered the wings associated with the cylindrical surface into two groups: one where parameters are estimated from fused data and one in which parameters are estimated from surface-only data. The mean parameter values and their standard deviations are reported in Figure 3.8. The 5 numbers associated with the cylindrical surface are the five cylinder parameters $(x_0, y_0, r_0, \alpha, \beta)$. Six and nine wings were detected with the limb (fused fits) and without the limb (surface-only fits), respectively. Of the 5 parameters, we only have the ground truth for r_0 , which should be close to 2.0. The fused fit performed better in having closer radius than the surface-only fits. We have no way of accurately measuring the ground truth of other pose parameters. By trying to fit the mean parameters from the surface-only fit as the initial estimates for fused fitting and vice versa, we found that the mean surface-only fit parameters are not good starting parameters for fused fitting (did not converge). However, the mean parameters from fused fitting are good starting values for the surface-only fitting. This result empirically suggests that the parameters recovered from fused fitting are more believable than those from surface-only fitting. In any case we are encouraged by the fact that the parameters clustered tightly.

By carefully studying the results, some general inferences can be made. We use Figure 3.8 to illustrate those inferences. We note that those inferences are true in general among all the images that were tested.

- 2D limb contour information is useful for recovery of surface parameters.

As stated above, parameters recovered from fused fitting should be closer to the ground truth. In Window A, the cylindrical surface parameters are recovered with $((+LN+CYL))$ or without $((+ +CYL))$ the limb contour. However note that the estimated radius is closer to the truth with fused fitting. In Window B, the surface-only fit $((+ +CYL))$ did not converge where as the fused fit did. Upon examining the initial parameter estimates, we found that the initial values of surface-only fits

are rather far from both sets of mean parameters described above. Note that the initial values for surface fits are derived from a local surface patch whereas the initial values for fused fits are derived with the aid of limb contour position in 2D. This leads us to believe that limb contour position in 2D is important information to use for recovering surface shapes.

- The program is able to correctly identify a cylindrical region near limb boundary from one that is not.

From sample output of window B, we see that the hypothesis that the cylindrical surface is near limb ((+LN+CYL)) is correctly accepted. In windows C and D, the hypothesis that the surface is near the limb is properly rejected. Only the hypotheses that the surface is *not* near limbs are accepted (e.g., (CYL+ +) and (CYL+PB+)).

- Curved contours can be correctly inferred.

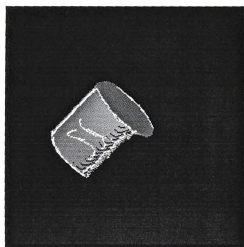
This is exemplified by windows D and E. where the correct local contour shape, parabolic, is correctly identified.

3.7.3 Open Cup vs. Lid Cup

In Figure 3.9 we demonstrate the power of wing representation and our wing detector. In Figure 3.9 (a), wing detection is performed on an image of a cup with no lid. The detected wings around the rim on top are correctly detected as jump wings. On the other hand, the wings around the rim on top of a cup with lid, as in Figure 3.9 (b), are correctly identified as crease wings. From the line drawing perspective, those two objects have the identical projection in the 2D. Because of the many-to-1 mapping in projecting 3D points into 2D image plane, line drawings in 2D often have ambiguous interpretations. This effect is also exhibited in Figure 1.2.



(a)



(b)

(a): (90 90) wing! - CYL
 (+PB+CYL 16.14)(CYL: x0=-25.09 y0= 39.21 r0= 1.31 alpha= 54 beta= 31)

(b): (90 90) wing! - CYL, PLN
 *(CYL+PB+PLN 17.74)(CYL: x0=-18.75 y0= 39.91 r0= 1.19 alpha= 55 beta= 25)
 (PLN: a = 1.61 b = -1.80 d = -33.90)

Figure 3.9. Wing representation can distinguish an opened cup from a cup with a lid. (a) Opened cup, the line corresponds to the rim is a jump edge. (b) Cup with lid, the line corresponding to the rim is a crease edge.

3.7.4 Summary on 10 Real Images

The summary of wing detection on the 10 real images depicted in Appendix A is given in Table 3.7.4. The window size for wing detection is fixed at 30 pixels \times 30 pixels. By carefully studying the table and the program output, we make the following observations.

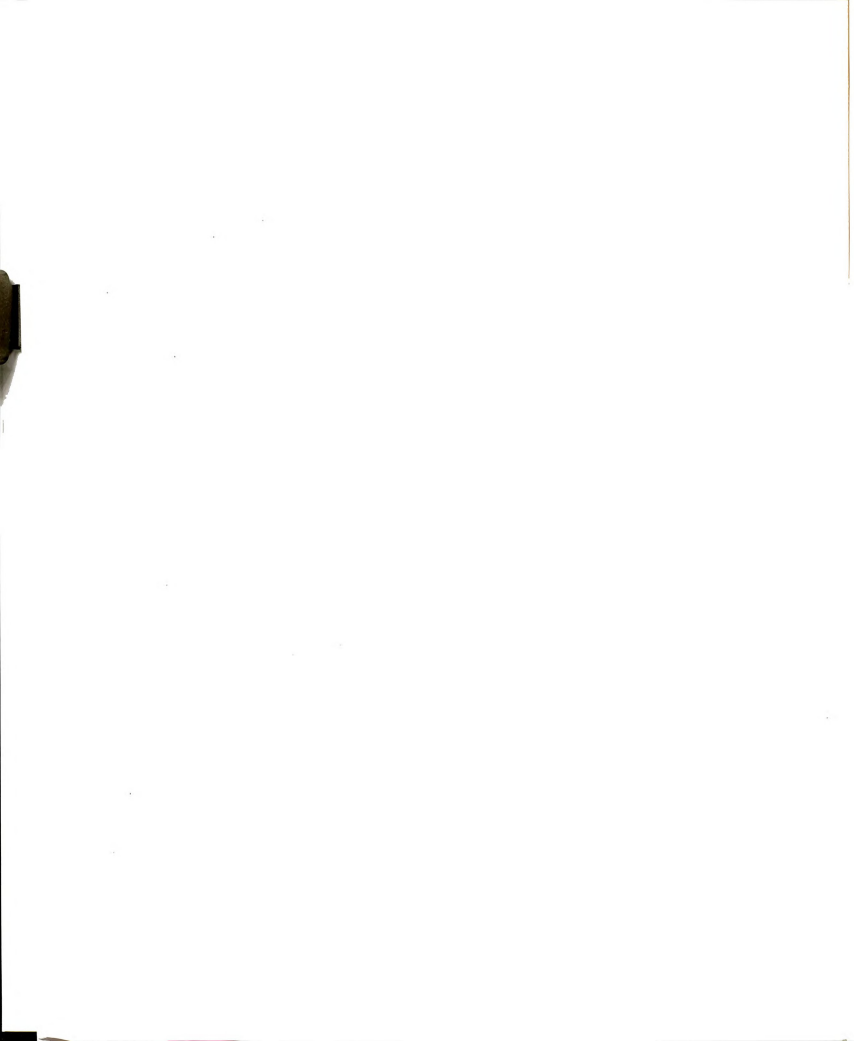
1. Planar surfaces are reliably detected, even when the visible surface patches are small. Furthermore, there are no *E4* errors. However, there tends to be more *E1* and *E2* errors. We believe a more intelligent busy window detector can remove most of the *E1* errors.
2. For fits involving spherical surface patches, the wings can also be reliably detected. We also noticed that the estimated radii tend to be smaller than the ground truth. There are also no *E5* errors in the two scenes with true spherical surface patches. The curved limb contour works strongly against the cylindrical hypotheses which must have straight limb contours. This agrees with our experiments done on synthetically generated data in Section 3.5.2.
3. Cylindrical fits were very sensitive to initial parameter values. We experienced better initial parameter estimations when the window overlapped the limb projection. The limb in 2D makes apparent the direction of the maximum curvature. Furthermore, it restricts the 2D projection of the axis to be parallel to it. Without the limb, the axis can only be estimated from the surface points, which is difficult if the patch is small. There were 13 *E3* errors in the 5 images involving cylindrical surface patches. All but one of those 13 errors occurred in windows that did not overlap a limb. Thus, as with the spherical case, the straightness of the limb boundary acts as a deterrent against spherical hypotheses.

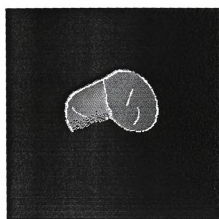
4. Finally we note that in the absence of the conical surface hypothesis, when fitting a conical surface patch the fitter will reject all the hypotheses when the surface patch overlaps part of the limb boundary. However, when the surface patch does not include the limb boundary, the spherical hypothesis will often be accepted. This again is in agreement with our finding with synthetic data.

3.7.5 More Wing Detection Examples

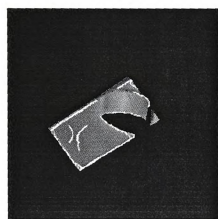
More examples of wing detection results are shown in Figure 3.10 (a)-(f). The figures show the detected wing samples overlayed on top of the original intensity images. Although some of the wings along an edge contour look ragged, this is partly due to the fact that short local edge contour may be well approximated by more than one curve type, resulting in possibly more than one wing of different wing contour shape being detected in the same window. Also, missing range data near edges causes holes in the displayed data. The current wing detector does not differentiate among those wings and *all* detected wings are returned for higher level processing. In the future, a smarter wing sensor could be designed to return only the "correct" wing for each window.

Note that there are some spurious wings and that some edge contours have no wings. On planar surfaces, spurious wings often have very similar wing surface attributes on either side and can be removed easily (see Section 5.3). For a spurious wing in a quadric surface, we observed that the wing surfaces often have different qualitative labels, making detection of it as a spurious wing difficult. This can be partly attributed to the fact that an interesting window is not divided evenly into two halves. If one of the regions becomes too small, an erroneous surface label may emerge from the non-linear least squares fit. In all the figures, edge contours with no wing can all be attributed to the fact that the fixed scale window always overlapped a fragment of those edge contours along with some other edge contours in the image,

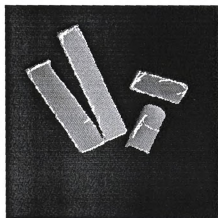




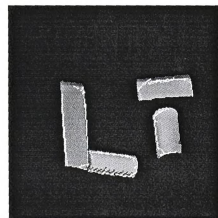
(a) cup



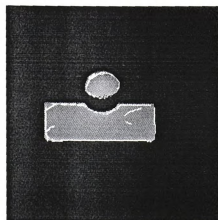
(b) hump



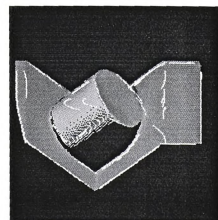
(c) block1+col1



(d) col1+col2

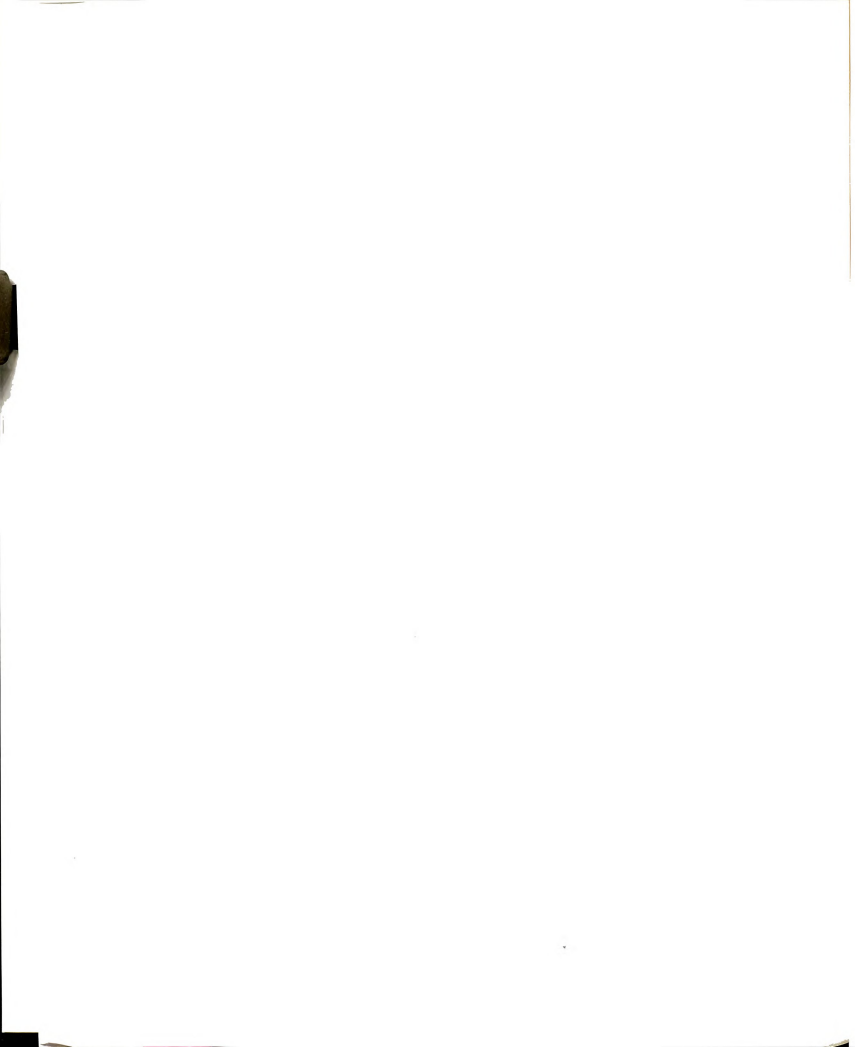


(e) block0+ball



(f) cup+jig

Figure 3.10. Wing detection examples of quadric surface scenes



which resulted in a busy window. In the future, a scaled-space wing detector may be tried to alleviate both of these problems.

The conclusions drawn from the previous examples also apply to these examples. The limb boundaries of the cylinder and the spherical ball are all correctly detected and labeled. Further, the surface parameters recovered with the limb contours are more consistent than those derived without it. More examples of wing detection on scenes consisting of only planar surfaces can be found in Figure 5.7.

3.8 Summary

In this chapter, a procedure for the detection of object wings in fused intensity-range data was outlined. Wings, which are local features, are detected from overlapping subimages. A simple busy/bland window detector is used to weed out the uninteresting subimages so that only those interesting subimages are further processed for wing detection. Hypotheses about the type of wings that are present are verified by fitting the hypothesized wing models against the data points in the subimage.

The novel idea of simultaneously fitting intensity contours and the adjacent range (surface) data is the central theme of the wing detector. We derived the contour equations of the limb projection of spheres, cylinders and cones from their respective 3D surface shape formulation and the projection of crease edges formed by intersecting quadric surfaces so that the 2D contour and 3D surface equation(s) are related by a common set of parameters. We showed by means of Monte Carlo experiments that (1) fused fitting results in fewer bad fits and more accurate parameter estimates than if surface or contour data were used alone; moreover, (2) fused fitting is better able to discriminate the different surface types. The advantage of fused fitting was more evident when higher Gaussian noise was introduced in the experiments.

We also studied the effect of different numbers of surface and contour points used for fused fitting. Contrary to popular belief, only a very small total number of data points are needed for the fit to achieve high success rate. This result seems to suggest that past fitting techniques that use hundreds and thousands of points are merely wasting valuable computing resources for a very small gain in confidence.

Wing detection over a large set of real fused images demonstrates the effectiveness of our wing detector. In general, correct wings are detected over "interesting" regions of the image and spurious wings, although present, are rare. The usefulness of the edge contours, especially limbs, in aiding surface fitting was apparent. It was shown that the limb contour is essential in generating good initial parameter estimates for wings near a limb boundary. We also observed that for curved surfaces, although the qualitative surface shape is often correctly inferred, the quantitative parameter estimates still need to be improved. The problem lies in generating good initial parameter estimates. Near a limb boundary, a more accurate guess can be induced with the help of its 2D position. Away from limb boundary, the initial estimates must be "guessed" over the small subimage area. Improving the initial estimate is a topic for future research. We also observed that if the region from which the surface points are sampled for fitting were too small, then the recovered qualitative and quantitative shape measures will often be incorrect.

Finally, we note some of the improvements that can be made to the wing detection algorithm. As is, wing detection is performed over a fixed scale; the window size is fixed for all images. Short image segments may never be recovered given the fixed window size and the image partition policy. To be robust, a scale-space approach can be used to prevent small features from being overlooked. Although this may increase the computational complexity, the algorithm is suitable for parallel implementation. Since wing detection within a window of subimage can proceed independently from the others, all wings can be detected concurrently on a parallel computer.

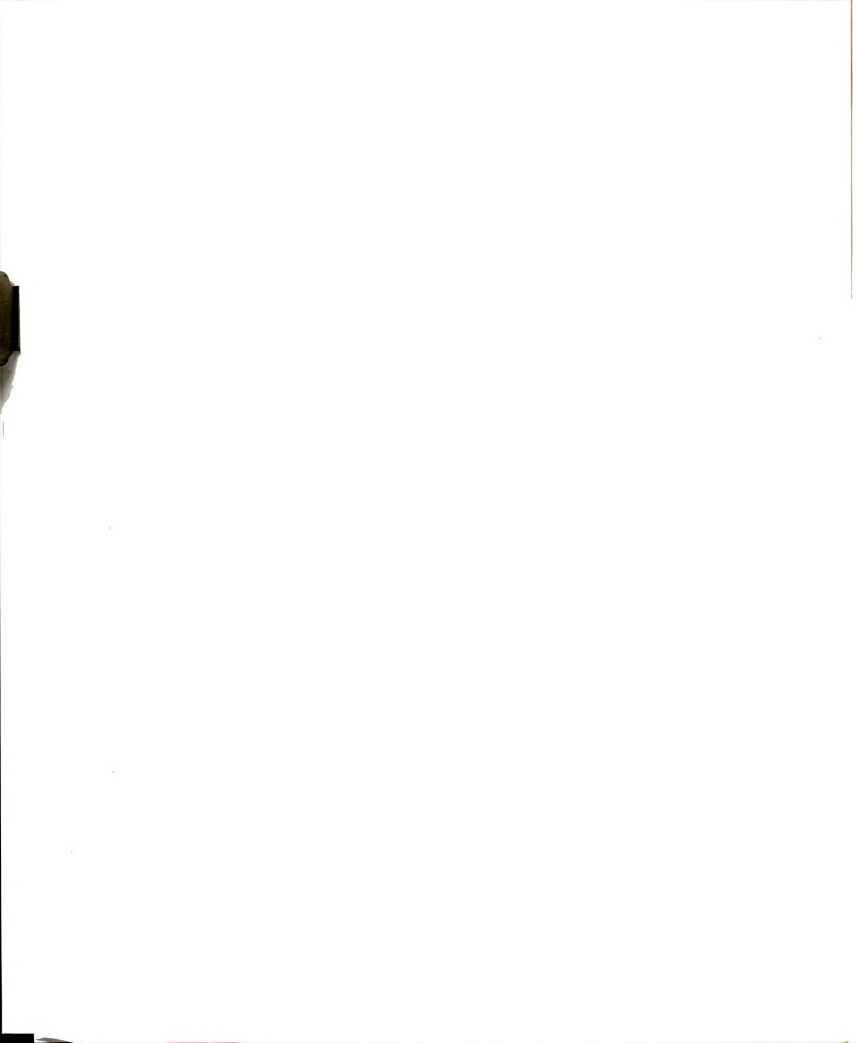
CHAPTER 4

Perfect Reconstruction of LLDG - Origami/Polyhedral World

4.1 Introduction

Line drawing analysis has been actively researched in the past. The goal of research in this field is to interpret the scene by interpreting the lines of the line drawing of objects derived from the scene. A common trait of the early papers is that they all assume that a perfect line drawing is available as input. They often left the difficult task of low level processing, taking a raw image and converting it to its line drawing representation to future research. In addition, a junction catalogue was usually derived in advance to facilitate the line interpretation process; this in turn placed unrealistic limitations on the domain of objects. The major drawback in those works is that by insisting on having the perfect line drawing in hand before interpreting the scene, one is faced with the difficult if not impossible mission of deriving the perfect line drawing from the raw data.

In this chapter, we will present two methods for complete construction of the labeled line drawing graph (LLDG) from a set of sampled wings under ideal assumptions. The added power of sparse range data is explored. With the benefit of the



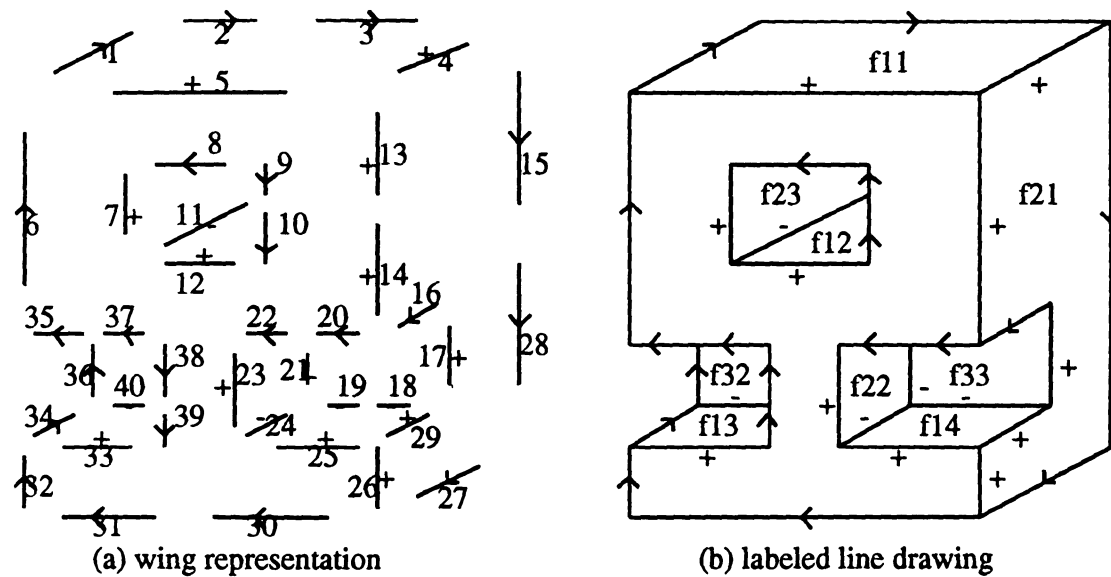


Figure 4.1. Wing representation and reconstruction of the Big-Block (from [26]).

range data, detected wings can be sorted so that data from individual planes can be interpreted separately. The basic idea is to first recover the line drawings of individual visible object faces; then the entire view is reconstructed by integrating the recovered individual faces. Two algorithms have been developed to do the reconstruction. In the first algorithm, global or local constraints are not necessary, provided that we can take sparse samples from the original range image. The second algorithm relies only on the geometrical properties of the LDG and does not need to reference the original range image. In so doing, not only will a perfect line drawing be constructed, but the lines and the junctions in the LDG will also be interpreted and the pose of each polygonal surface in the scene will be recovered (the labeled line drawing graph). An example showing the wing representation and reconstruction of an object is depicted in Figure 4.1.

Both algorithms to be presented in this chapter can be classified as deterministic algorithms because no heuristic procedures are involved in the reconstruction process. This is only possible due to the strong assumptions made about the completeness of

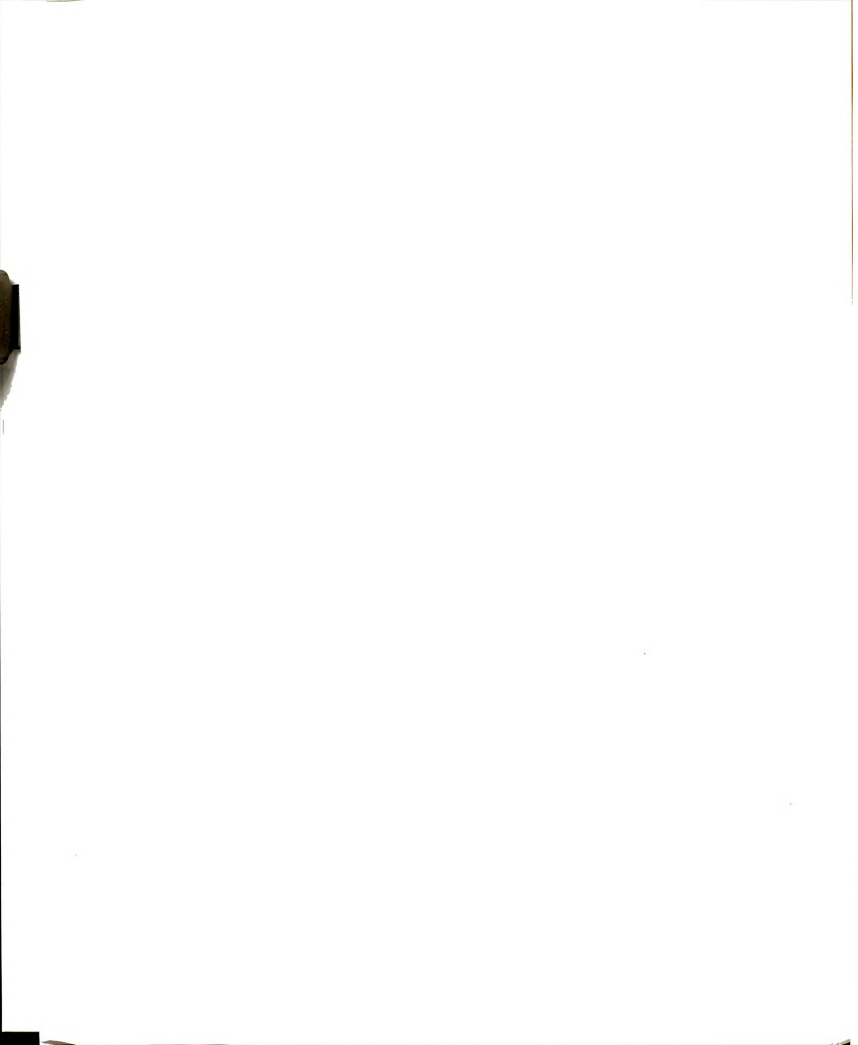
the input wing set. A heuristic based algorithm which relaxes some of the strong assumptions will be discussed in the next chapter. The main differences between this research and that of the past are that: (1) a perfect line drawing is derived instead of assumed; (2) a junction catalogue is not needed in interpreting the line drawing; and (3) surface pose information is also recovered.

In the next section, we will survey some of the previous research done on line drawing analysis. This is followed by the presentation of the two LLDG reconstruction algorithms. Arguments for correctness of each of those algorithms will be given. Section 4.5 provides reconstruction examples from both syntactically generated scenes and real scenes. A section summarizing the results and contribution of this research concludes the chapter.

4.2 Survey of Related Background

Analysis of line drawings have received much attention since the early days of computer vision. Interpretation of line drawings of polyhedral objects has been studied quite extensively [30, 61, 66, 82, 108, 115, 123] and is well understood. Only recently, have works on line drawing interpretation been extended to a limited class of curved objects [83, 92, 93]. One common drawback of all those algorithms is that they all assumed that a perfect line drawing can be acquired from the input image. However, to date, no edge detection algorithms can yield perfect line drawings so that those algorithms can be applied.

In the following subsections, past work in the area of line drawing analysis will be reviewed. Most works have concentrated on the polyhedral world. Only recently has interpretation of curved line drawings received any attention. One significant result coming out of those studies is that a curved line in the image may be labeled as an occluding edge in one part and a crease edge in the other [83].



4.2.1 Model Based Interpretation of Line Drawings

One of the first significant works on interpreting two dimensional line drawings of three dimensional objects is due to Roberts [108]. The main component of his 3D object recognition system is the database of models of objects to be recognized. Given an image of an object from the model database, the *perfect* line drawing of the object is extracted from the image. The database is searched for a model whose perspective projection, after some translational, orientational transformation and scaling, coincides with the extracted line drawing. Although this system required the strong assumptions that (1) a perfect line drawing can be extracted, and (2) objects are isolated in the images, it did pave the way for future research on line drawing interpretation and object recognition via a model database.

Falk [34] also used fixed models of the objects that would appear in the scene to help in identifying the visible objects in a photograph of a scene and determining their orientation and positions in 3D space. There are two main differences between the systems of Falk and Roberts. For one, Falk allowed for imperfect input (line drawings generated from the photograph can be degenerated views of the objects or some edge may be missing all together) as opposed to the perfect line drawing required by Roberts. Secondly, Falk's models specified precise shapes and sizes as opposed to Roberts' models, which are generic in the sense that a cube can represent any right parallelepiped. A hypothesize-and-test strategy was used by Falk in identifying and locating objects in the scene.

4.2.2 Model-Free Interpretation of Line Drawings - Polyhedral and Origami World

In contrast to Roberts, Guzman [51] did not attempt to recognize isolated objects in a scene. Instead, he developed a program called "SEE" which took a perfect line

drawing of a scene consisting of jumbles of solid polyhedra with occlusion and tried to partition it into a set of individual objects. The partitioning was based on a set of *ad hoc* rules where configurations of lines and regions at junctions dictate how the regions are to be decomposed/grouped. Although this method worked well for many complicated line drawings, it was nevertheless a heuristic algorithm. No justification of why the rules are correct was given. Examples of line drawings where the algorithm may fail were given by Mackworth [82].

The first systematic scene interpretation based on line drawing labeling was largely credited to Huffman and Clowes [30, 61]. They independently went about line drawing interpretation of trihedral polyhedral scenes. A trihedral polyhedra is an object in which exactly three planar surfaces meet at each vertex. Each line in the line drawing can be labeled by exactly one of four symbols: $\{+, -, >\}$. A “+” indicates projection of a convex edge where both surfaces are visible; a “-” indicates projection of a concave edge where both surfaces are also visible; and a “>” indicates projection of an occluding edge where only one of the two surfaces is visible and the visible surface lies to the right as one moves in the direction of the arrow. Junctions which correspond to visible vertices are labeled as a combination of line labels of all lines meeting at each junction. Assuming a general viewpoint, an algorithm can be formulated to systematically, yet exhaustively examine all physically realizable junctions. Given the trihedral assumption, the three faces of any vertex define three intersecting planes, which divide the space into 8 octants. By considering all ways of filling up these 8 octants with object material and viewing the vertex from un-filled octants, a complete junction catalogue can be obtained. A complete listing reveals that only 18 of 208 possible junction labels are physically realizable (see Figure 4.2).

Using the junction catalogue, valid interpretations of line drawing of trihedral world are rendered through exhaustive search. This labeling scheme is not without problems. More than one interpretation of a single line drawing is possible (Fig-

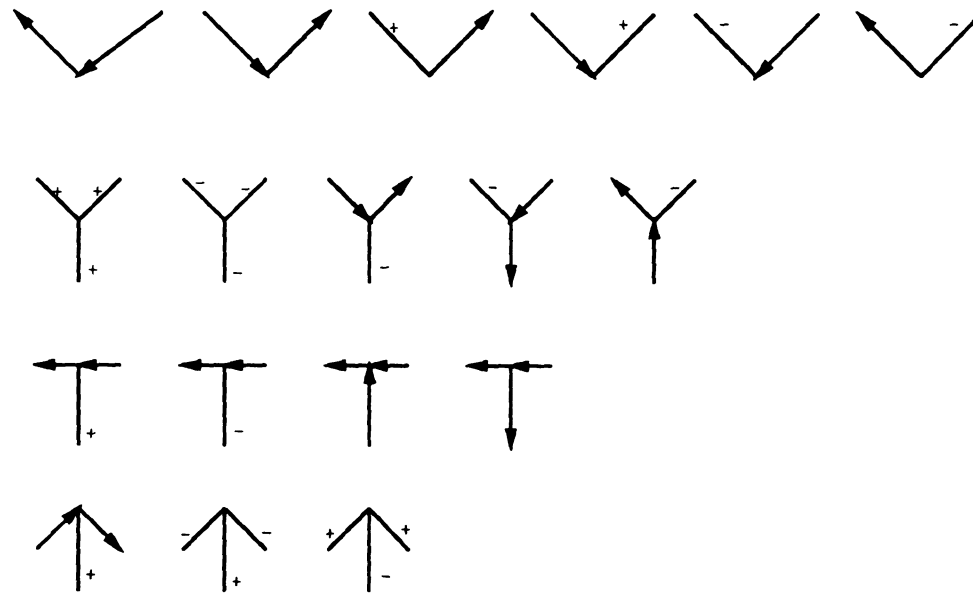


Figure 4.2. Huffman-Clowes junction catalogue.

ure 4.3 (a)). Moreover, nonsensical scenes are coherently labeled (Figure 4.3 (b)); line drawings may be given geometrically impossible line labels (Figure 4.3 (c)); and scenes that cannot arise from polyhedra are easily labeled (Figure 4.3 (d)).

Waltz [123] extended Huffman and Clowes' research in two important ways. One, he expanded the Huffman-Clowes set of line labels to include shadows, cracks, and separable concave edges. As a result, the number of line labels increased from 4 to 11 and the junction catalogue was greatly expanded due to the addition of 4- and 5-line junctions. Approximately 2593 junction labels were physically possible. It is clear that exhaustive search for a consistent line labeling as with Huffman-Clowes is computationally prohibitive. The second contribution from Waltz is that he replaced the exhaustive search for consistent line labelings by a constraint-propagation algorithm to weed out impossible junction labels at each junction. A direct tree search followed to enumerate all possible labelings. However, the problems associated with Huffman-Clowes labeling still persist.

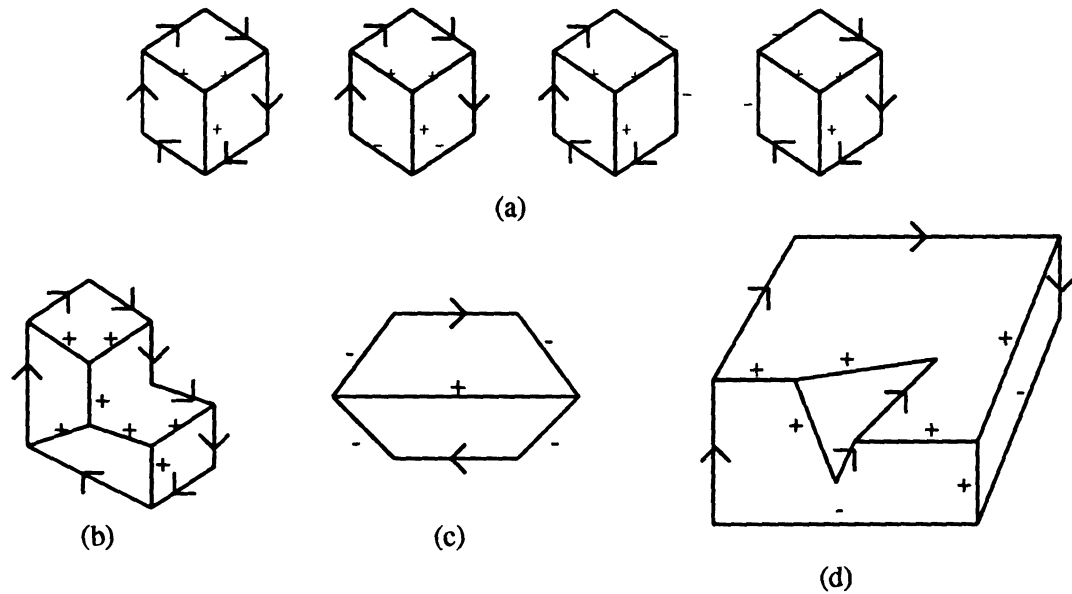


Figure 4.3. Problems with Huffman-Clowes labeling scheme. (a) More than one interpretation may exist for a single line drawing. (b) Syntactically nonsensical scenes are coherently labeled. (c) Line labels that are geometrically impossible (d) Non-polyhedra can also be labeled (b, c, d are from [7])

To deal with the problem of legal labelings that are not realizable as polyhedra, Mackworth [82] proposed the use of gradient space to remove those labelings. Simply stated, the gradient (p, q) measures the instantaneous change in the depth of a surface at point (x, y) . The gradient space is then a representation for vector orientations and is used to represent the surface normals of the polyhedral faces. For a planar surface, all points on the plane map to the same (p, q) point in the gradient space. Using the same set of line labels as Huffman and Clowes, Mackworth was able to interpret line drawings by reasoning about surface orientations based on the properties of the gradient space. Not only were the labelings which yield geometrically impossible objects removed, but also the object domain was broadened to include all polyhedra (not just trihedral polyhedra).










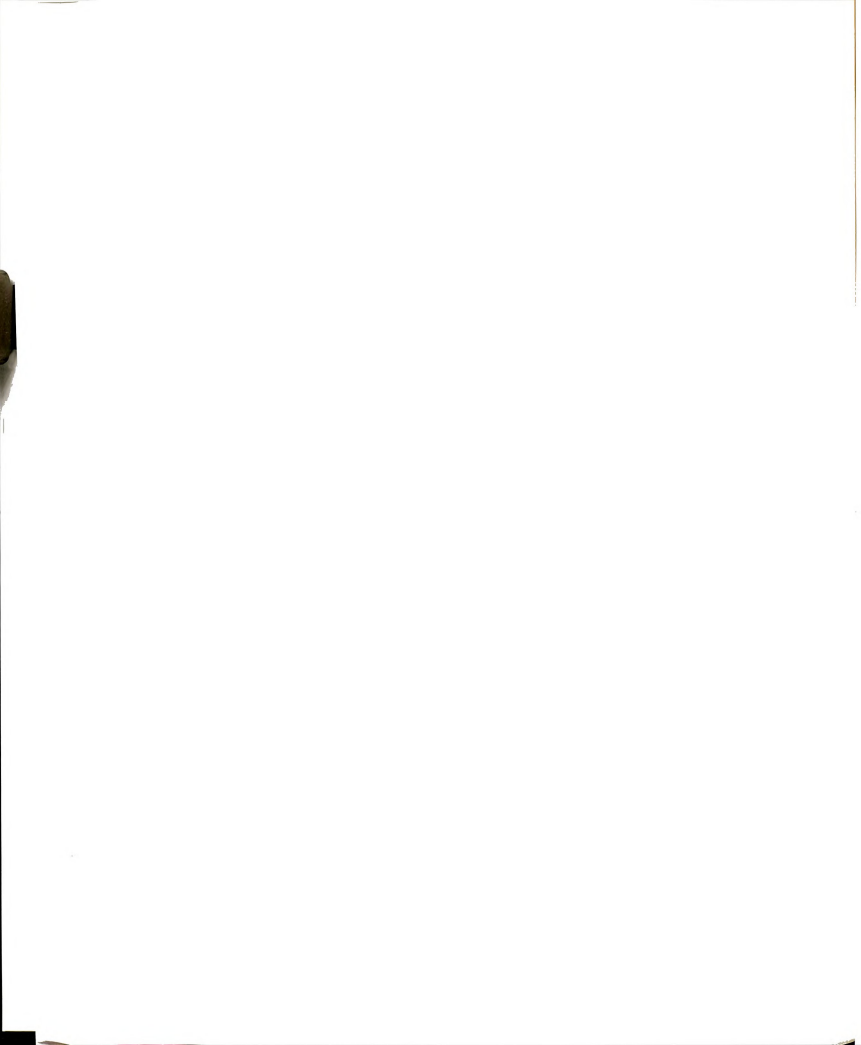
	L	8		X	7
	ARROW	15		PSI	22
	FORK	9		XK	41
	T	16		AST	16
	K	19			

Figure 4.4. Origami junction types and number of possible junction labels for each type (from [66]).

Mackworth's approach was continued by Kanade [66]. Kanade broadened the object domain of Huffman and Clowes to include origami objects. The assumptions were that surfaces in the origami world are assumed to be planar, that no more than three surfaces of different orientations meet at a vertex, and that no more than three edges of different orientations are involved at a vertex. Orthographic projection was assumed. Under this set of assumptions, a new junction catalogue was derived much like that of Huffman and Clowes. Line labels were still restricted to the set $\{+, -, >\}$. However, 9 junction types are now possible (Figure 4.4) and the total number of entries in the junction catalogue jumps from 18 to 153. Line labeling proceeds by using a filtering procedure for labeling junctions similar to that of Waltz and checking the consistency of surface orientation. Kanade later introduced two heuristics to filter out unnatural interpretations [67].

Sugihara [115] took the algebraic approach to line drawing interpretation. First, a set of candidate spatial interpretations was generated for the given line drawing via constraint propagation much like that of Waltz [123]. For each interpretation, spatial information such as the relative depth between a pair of vertices or a vertex and a



surface was derived from the line label. The task of determining the correctness of each candidate interpretation was reduced to the problem of testing whether a system of linear constraints has a solution or not.

Assuming that there are m (r_1, r_2, \dots, r_m) regions and n (j_1, j_2, \dots, j_n) junctions in the line drawing, the system of linear constraints was derived as follows:

- (1) Each region r_i is associated with a 3D planar face f_i with linear equation $a_i x + b_i y + z + c_i = 0$. Each 3D vertex v_α which corresponds to a junction j_α and which lies on f_i gives rise to a linear equation $a_i x_\alpha + b_i y_\alpha + z_\alpha + c_i = 0$. Note that the unknowns are a_i, b_i, c_i and z_α . A system of linear equations is obtained by collecting all such equations, one for each junction in each region.
- (2) A system of inequalities can be gathered from the relative depth of all pairs of vertices corresponding to junctions on the line drawing. If a vertex v_i is in front of v_j then

$$z_i < z_j$$

else

$$z_i > z_j.$$

- (3) Finally, a system of inequalities are formed using the relative depth information between a vertex v_α and a surface f_j that correspond to a junction j_α and a region r_j on the line drawing. If v_α is in front of f_j then

$$a_j x_\alpha + b_j y_\alpha + z_\alpha + c_j > 0$$

else, if v_α is behind f_j then

$$a_j x_\alpha + b_j y_\alpha + z_\alpha + c_j < 0.$$

Appending the above three systems results in one big linear system of equations and inequalities with $n + 3m$ ($z_1, z_2, \dots, z_n, a_1, b_1, c_1, \dots, a_m, b_m, c_m$) unknowns. Sugihara showed that the existence of a solution to this system of linear equations and inequalities is a necessary and sufficient condition for a labeled line drawing to represent a valid planar surface scene.

4.2.3 Model-Free Interpretation of Line Drawings - Curved Surface World

Turner [121] attempted to extend the trihedral vertex polyhedral junction catalogue to smooth faced opaque solids. The surfaces were composed solely of planar, parabolic, elliptic, or hyperbolic points and bounded by surface-normal discontinuous edges. A general viewpoint was assumed and shadow boundaries were allowed. Several other assumptions about the viewpoint, background, illumination, and corners were also made. However, unlike the junction catalogue for trihedral vertices, Turner's catalogue was not shown to be complete for its domain [91].

Chakravarty [25] dealt with planar-faced and curved-surface solids, having vertices formed by at most, three surfaces, and edges formed by two surfaces. Seven generalized junction types were defined. Limitations on permissible junction types as one moves from one end of a line segment to the other provided the ability to verify whether a given sequence of junctions forms a realizable configuration. This labeling scheme dealt with regions and lines, rather than with the geometric characteristics of a junction.

Malik [83] systematically derived a junction catalogue (Figure 4.5) for scenes comprising opaque regular solids bounded by piecewise smooth C^3 surface patches with no markings or texture on them. Orthographic projection and a general viewpoint were assumed. The input line drawing to the interpretation module was assumed to

be perfect and that lines due to shadows or specularities were not present. An algorithm which utilizes this catalogue to determine all legal labelings of the line drawing was developed. A new line label \ll was introduced to accommodate *limb* boundaries, a locus of points on the surface where the line of sight is tangent to the surface (see Figure 4.6). In order to prune highly counter-intuitive interpretations involving a number of hidden faces, a local minimum complexity rule was also used to restrict the number of hidden faces at a vertex to at most one. Although a large number of curved objects were allowed in the object domain, simple objects such as the piece of nose cone shown in Figure 4.6 were excluded.

A mathematical framework for line drawing interpretation was recently presented by Nalwa [91]. The assumptions were that the imaging geometry is accurately captured by orthographic or perspective projection, the viewpoint is general, the surfaces are piecewise C^3 , and that limbs are the only viewpoint-dependent edges. Using this framework, constraints on the scene from instances of straight lines and conic sections in line drawings were derived [92]. In [93] constraints associated with bilateral symmetry were investigated. It was shown that the orthographic projection of a surface of revolution exhibits bilateral symmetry about the projection of the axis of revolution, irrespective of the viewing direction. However, no attempt was made to suggest schemes for the detection of bilateral symmetry in line drawings.

4.3 Reconstruction of LDG Via Resampling

An LLDG reconstruction algorithm similar to the one presented in this section was first conceived by Chen [26]. However no implementation was attempted and there were more restrictive assumptions. Here, we present a slightly different algorithm that does not require the objects to be completely inside the field of view and show the results of our implementation.

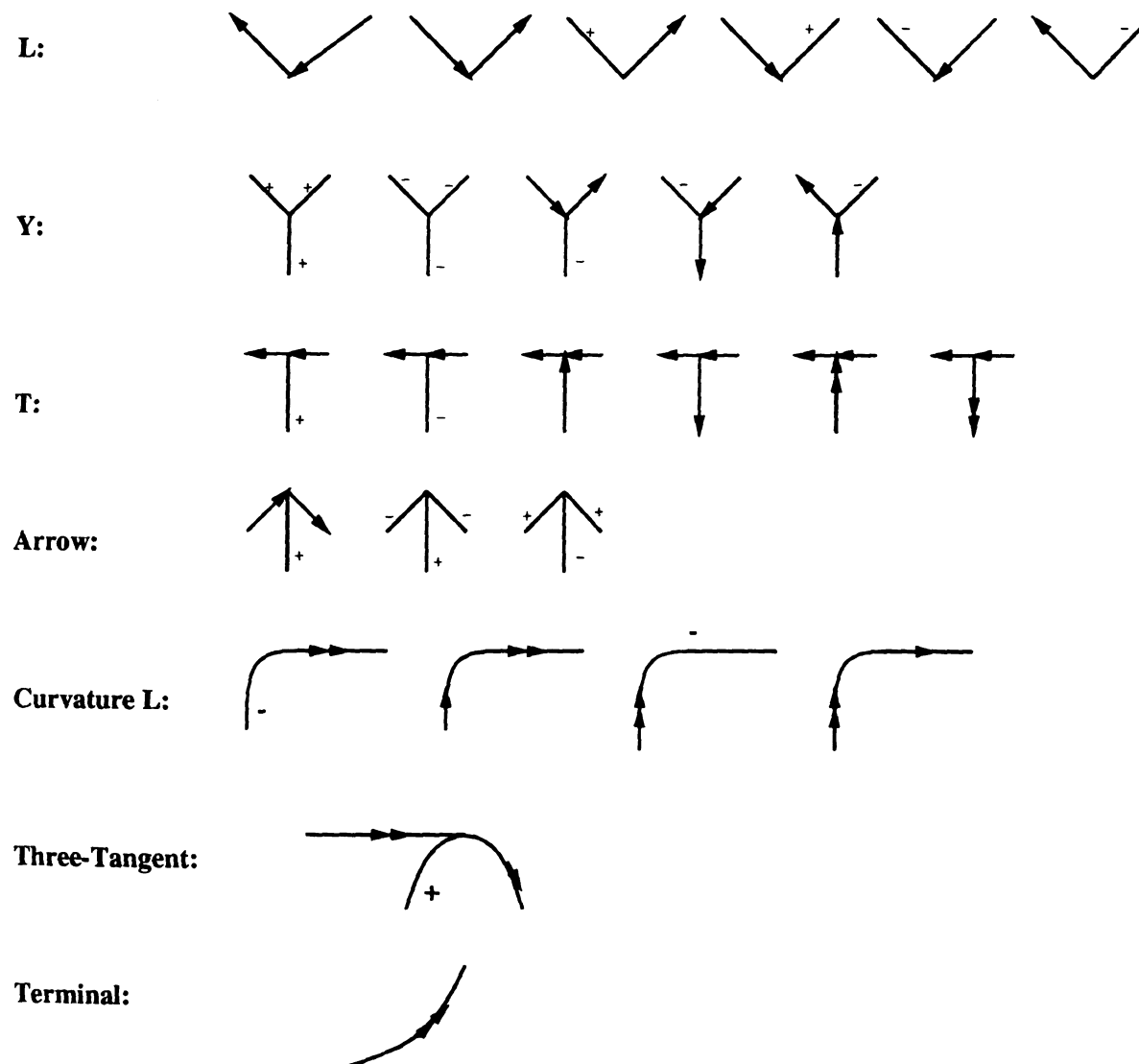


Figure 4.5. Junction catalogue for piecewise smooth surfaces [83].

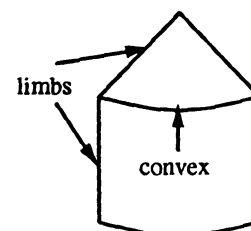
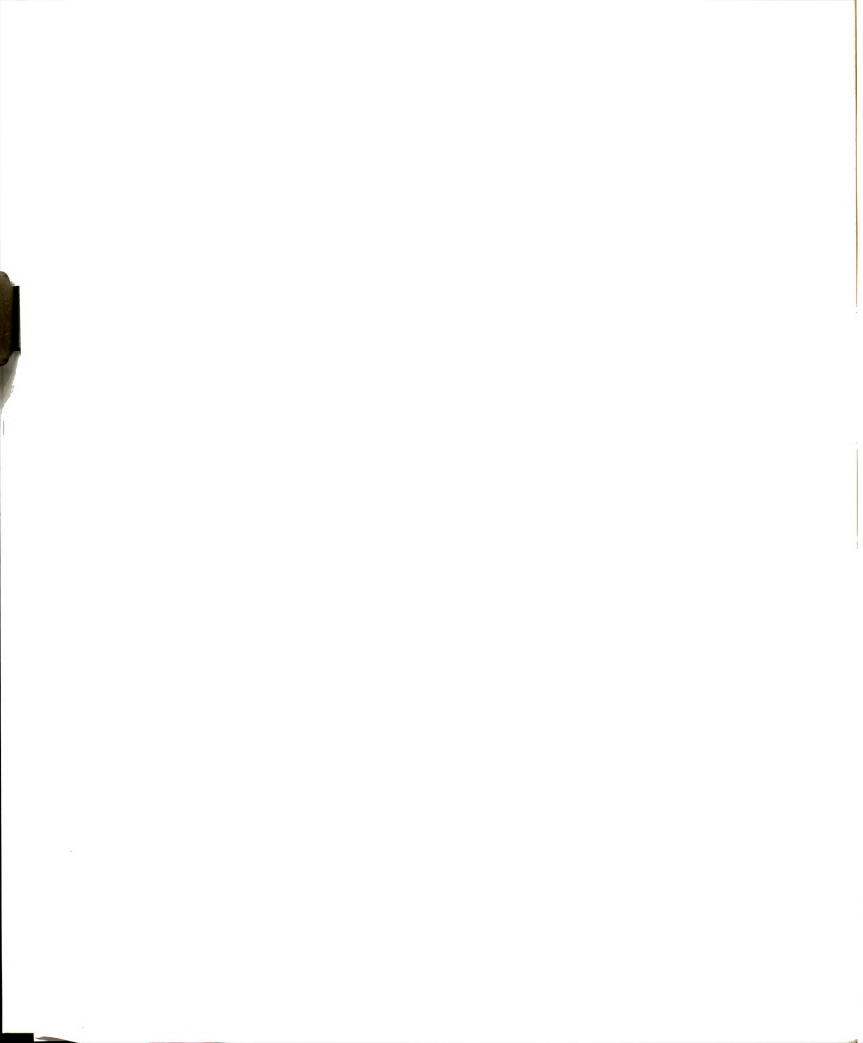


Figure 4.6. Apiece of nose cone is not in Malik's [83] object domain.



4.3.1 Assumptions

The reconstruction algorithm depends upon the following assumptions.

Object Domain/Scene Assumptions:

- A_1) Objects are δ -polygons or polyhedra whose faces are δ -polygons.
- A_2) Scenes can be composed of multiple objects which may occlude one another.

Wing Detector Assumptions:

- A_3) The view is non-accidental and the "wing sensor" produces at least one wing sample for each edge that would be visible in the LDG. (Recall the definition of a non-accidental view from Section 2.3.)

- A_4) There are no spurious wings.

Computation Accuracy Assumption:

- A_5) Measurements and computations are infinitely accurate.

Clearly assumptions A_3 , A_4 and A_5 are unrealistic in lieu of existing low level image processing techniques. However, our intention is to present a deterministic algorithm such that a complete line drawing graph can be reconstructed under these ideal assumptions. In the next chapter, we shall give a heuristic algorithm for reconstructing LDGs that relaxes these assumptions. An object view and its wing representation which satisfy these assumptions is given in Figure 4.1.

4.3.2 Algorithmic Approach

Our approach is to individually reconstruct the LDG of each face of the objects in the view then merge them to form the final LLDG. For each plane, the line segments from

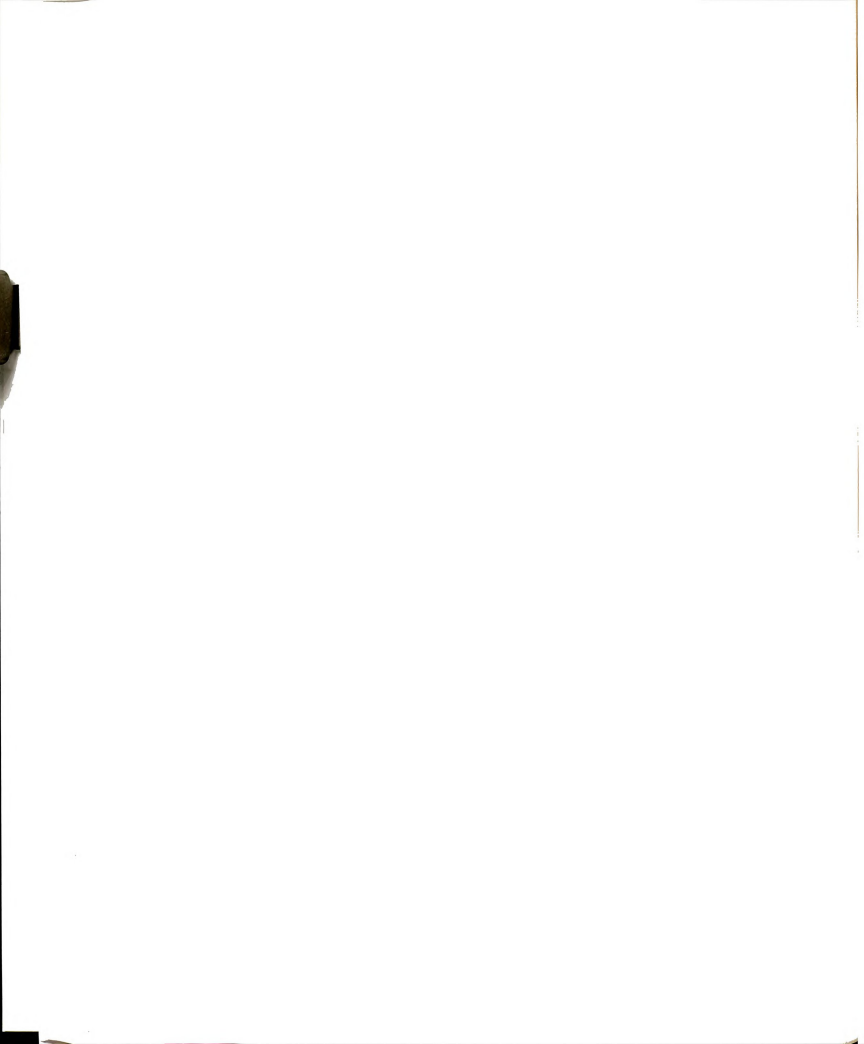
all wing samples involving that plane are extended until they intersect the image frame forming the set of all possible LDG line segments. When necessary, the algorithm resamples the range data to decide whether one of the possible segments is really in the LDG. The example below originates from Chen [26].

Step 1 : Cluster wings into plane groups

Wings are clustered into groups such that wings in the same group lie on the same plane. Counting background as one planar group, each wing will be a member of exactly two plane groups. From Assumption A_3 , the wing sensor produces at least one wing for each LDG segment. Thus, all planes containing object faces will be known as well as all lines containing visible edges. In addition, from Assumption A_4 , there can be no spurious wings; hence no spurious planes. Consider the example in Figure 4.1 (a). After grouping wing segments according to their plane equations, 8 plane groups are obtained. Processing of one plane group, { 4, 13, 14, 15, 16, 17, 26, 27, 28, 29 }, from the right face of the block will be continued below. Note that it is possible for a planar group to be composed of wings from two or more faces. For example, wings from faces f_{13} and f_{14} or f_{32} and f_{33} in Figure 4.1 (b) belong to the same group because the two faces are coplanar. The following steps show how to reconstruct LDG segments from the wings of a single plane group. At the very last step, all labeled segments will be merged to complete the interpretation of the LDG.

Step 2 : Determine all possible LDG segments of a planar group

For each wing segment in a plane group, construct a straight line within the picture frame (see Figure 4.7 (a)). This will result in a set of junctions and line segments which represent the possible images of the 3D object features (vertices and edges). From Assumption A_5 , collinear wing segments will lie exactly on the same constructed line. There are no extra lines generated (Assumption A_4) and no missing lines (As-



sumption A_3). The object edges visible in the LDG must form a subgraph embedded in the planar graph of all these possible line segments: it is the goal of the next steps of the reconstruction algorithm to extract the correct subgraph (or subgraphs in case of ambiguity).

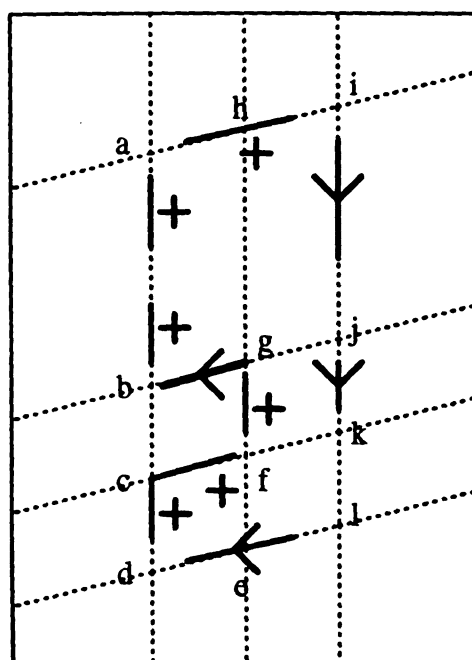
At any time in the reconstruction, a line segment can be classified into one of the following three categories: a *must*-segment, an *undecided*-segment or an *unwanted*-segment. A *must*-segment is a line segment that has been determined to be part of an object face boundary. An *undecided*-segment is one that is not a *must*-segment, but has not been discarded. An *unwanted*-segment is one that has been determined to not be part of a face boundary. We now select only those line segments which form the visible boundaries of the given 3D object faces in the given plane.

Step 3 : Determining initial must-segments

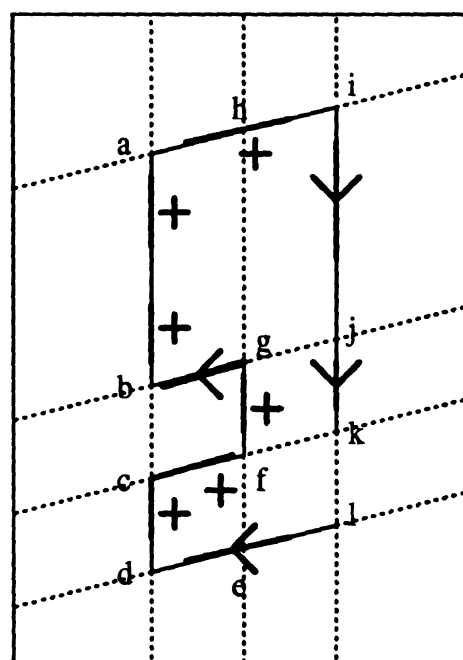
Because of Assumption A_4 , a line segment overlapping a wing segment must indicate some object edge and hence should become a *must*-segment. The result for the right face of the block in Figure 4.1 is shown in Figure 4.7 (b). There are also several undecided line segments, denoted by broken line segments in Figure 4.7 (b). For example, LDG segment \overline{kl} has not yet been recovered because no wing segment is associated with it. Such segments are examined in the next step to decide whether or not they should be part of the LDG.

Step 4 : Access the range data for all undecided-segments

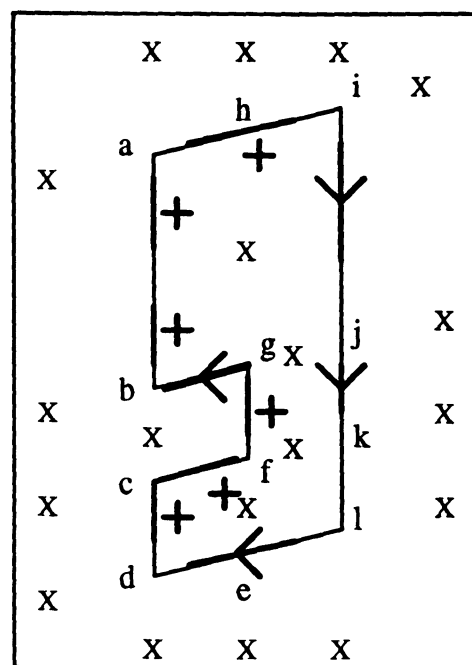
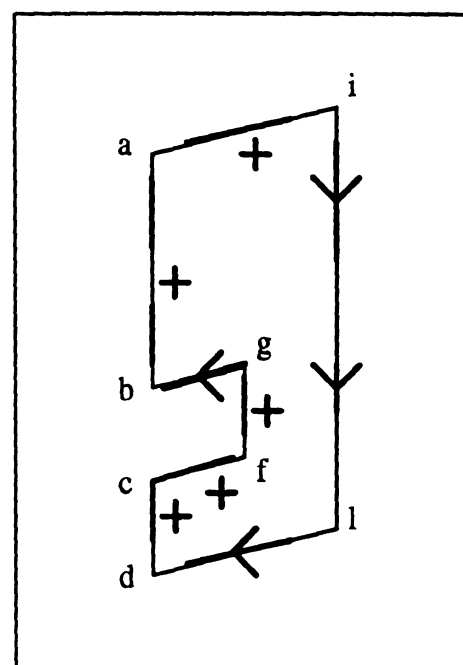
Any line segment in the LDG must separate exactly two distinct regions. If 3D points projecting to both sides of a candidate line segment are located on the same 3D plane, then the undecided line segment can be deleted from consideration. An undecided segment is unwanted unless it has exactly one side with points on the 3D plane of the current wing group. Assumptions A_1 and A_2 ensure that there is sufficient surface



(a) Step 2: all line segments extended.



(b) Step 3: line segments overlapping sensed wings are must segments

(c) Step 4: range sampling removes 19 unwanted segments(x) and includes must segment \overline{kl} 

(d) Step 5: delete redundant intersections.

Figure 4.7. Deciding on LDG line segments (Figure originated from [26]).



in the 3D view to sample and make this decision. By resampling the range data on either side of the undecided-segment, every undecided line segment can be classified as a must- or unwanted-segment. Thus unique LDG is obtained for each planar group of faces. Returning to the running example, the line segment between k and l in Figure 4.7 (b) can be successfully recovered, while all other undecided-segments, including \overline{bc} , \overline{cf} , \overline{gh} , \overline{gj} and \overline{fk} , are identified as unwanted-segments. The result of this step is depicted in Figure 4.7 (c).

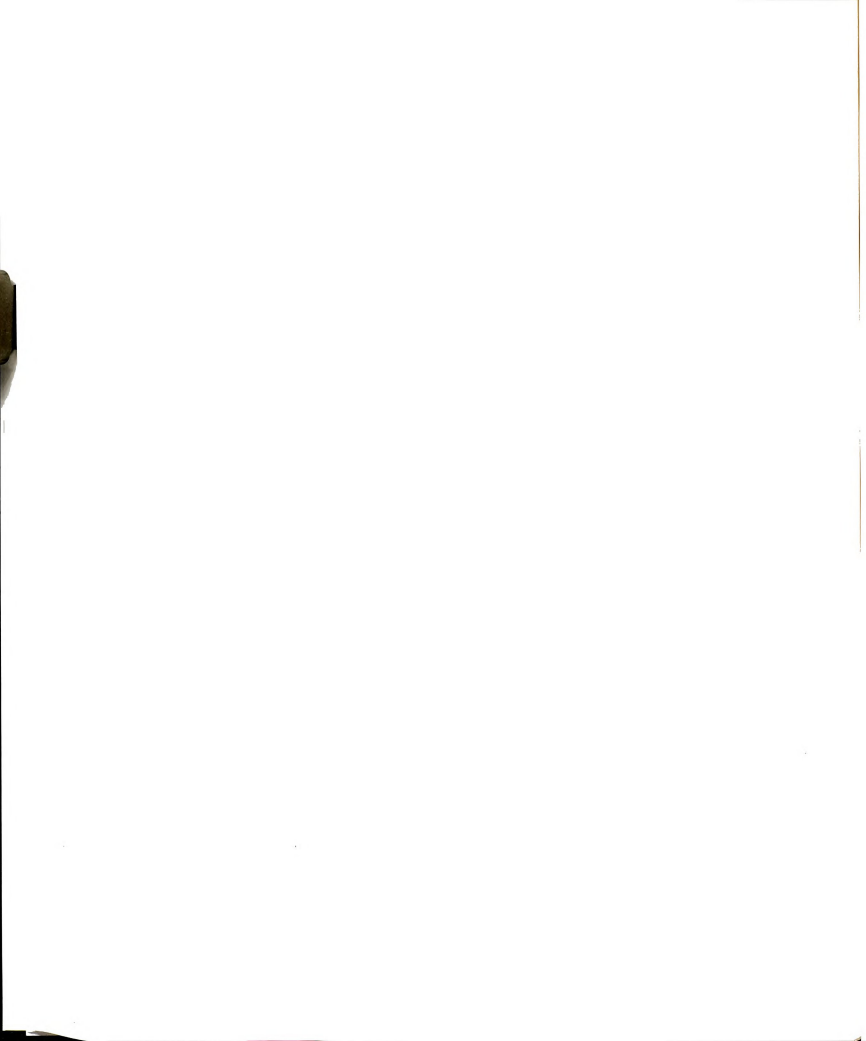
By using this resampling rule, each of the undecided segments is either kept as a must-segment or discarded as an unwanted-segment. Obviously if the range image is not available for resampling, this step would not be possible. In the next section, we give another way of reconstructing faces from each planar group using rules derived from geometric constraints instead of resampling range data.

Step 5 : Delete redundant junctions.

Returning to Figure 4.7 (c), we see that junctions e , h and k do not correspond to vertices of a 3D polygon. From object and view point assumptions all regions in the image plane are ϵ -polygons. Thus each junction of the newly reconstructed region must be shared by an even number of line segments not all of which are collinear. Therefore, after removing unwanted line segments, junctions which become redundant are removed. After applying steps 2 to 5 to each plane group, the following conditions must hold for each group: every junction has even degree and line segments form a set of circuits. The distinct object faces from a given plane group are represented by distinct sets of circuits.

Step 6 : Merge planar groups to form the LDG.

Up to this point the algorithm has recovered all edges and vertices of polygonal faces. The remaining work is to reconcile the line segment labels for the entire LDG. T-



junctions are created when plane groups are merged and line and sub-line segment labels are assigned according to the recovered T junctions and the wings associated with the given line segments. Figure 4.1 (b) shows the result of such a process. More interesting is the analysis of the “table & chair” in Figure 4.19 : when planar face $ABCD$ is reconstructed, points G and H are not known – they are formed by the merger with other faces. Note that the junction H is not in the Huffman-Clowes catalogue. Junctions (vertices), such as C and F , are formed by merging 2D (3D) points of intersection computed from the original wing data : assumptions A_1 and A_2 give us tolerances to use for this task. (Also note that the geometric character of \overline{DA} can be determined by the wing sensor to be one of $\{ + \text{ (chair-top collinear with table edge)}, < \text{ (chair-top closer than table top)}, > \text{ (chair-top occluded by table-top)}\}$, but in no case can the connectivity of the materials being sensed be determined without fallible heuristics being applied on top of the LDG construction.)

4.3.3 Resampling Reconstruction Algorithm: POLY-1

The above steps are summarized below in an algorithm skeleton. The input includes the range image and an extracted wing representation of a polygonal scene. The output includes a labeled line drawing, the equations of visible faces and the coordinates of visible vertices. Examples of scenes that can be handled are given in Section 4.5.

- (1) Cluster wing segments by plane equation
- (2) For each plane group
 - (2.1) generate a line for each wing
 - (2.2) compute the intersections among all lines
 - (2.3) preserve line segments which overlap a wing
 - (2.4) remove or preserve line segments by testing the range data
 - (2.5) delete redundant intersections
- (3) Merge LDGs of individual polygonal faces

- (4) Compute 3D vertices by intersecting line equations
- (5) Retain all line segment labels from wings
- (6) Output LDG with plane equations, 3D vertices and edges

4.4 Reconstruction Via Geometric Constraints

While the algorithm developed in Section 4.3 is able to reconstruct a unique LLDG from the wing samples, it requires the original range image to be available for resampling. However, the range image may not be available in some situations. Furthermore, future research may enable wing detection without range data. Thus, an algorithm for the recovery of the LLDGs using only the set of sampled wings is desirable and is developed in this section. The algorithm is similar to the previous one except for the way individual planar groups of wings are reconstructed into regions. Rather than resampling the range data to complete the line drawing of each of the coplanar faces, we show that the LDG can be reconstructed using the assumptions and the inherent geometric constraints of the projection of polygonal scenes. Thus only the region reconstruction step will be described in this section.

Given a polygonal scene consisting of only objects in the object domain and a non-accidental view of the scene, assume there are m visible faces which lie on n , $n \leq m$, different planes, the task at hand is the reconstruction of the line drawing of the faces on each plane. Let G denotes the perfect LDG of the faces on one polygonal plane as viewed through the given non-accidental viewpoint. G is not the LDG of the scene; rather, the LDG of a set of coplanar δ -polygons in the scene. In the following subsections, we will exploit the geometrical constraints on G and formulate reconstruction rules base on those constraints.

4.4.1 New Terminology

To aid the understanding of the geometric constraints and the new algorithm, we define the following functions such that each returns some specific information about a particular line segment, junction or region label. The first four functions are similar to those of Baumgart [9] but are renamed here to better suit our data structure.

Denote the regions (ϵ -polygons) to which the faces (δ -polygons) project as $\{R_1, R_2, \dots\}$, and the line segments in the LDG as $\{seg_{ij}, seg_{jk}, \dots\}$ where seg_{ij} denotes the segment from junction v_i to junction v_j . Furthermore, the regions $\{R_1, R_2, \dots\}$ all have the same planar label, say P_m , which is the equation of the plane from which the line drawing of object faces are being reconstructed. Note that due to occlusion, there may be more junctions in the LDG than there are vertices in 3D.

$SegCCW_{v_i}(seg_{ij})$: returns seg_{ik} where seg_{ik} is incident to junction v_i and is the first segment encountered from seg_{ij} in the *counter-clockwise* direction. (same as PCW(edge) in [9])

$SegCW_{v_i}(seg_{ij})$: returns seg_{ik} where seg_{ik} is incident to junction v_i and is the first segment encountered from seg_{ij} in the *clockwise* direction. (same as NCCW(edge) in [9])

$RL-CCW_{v_i}(seg_{ij})$: returns region label P_δ of the region encountered in the *counter-clockwise* direction from seg_{ij} with viewing origin at v_i . (same as PFACE(edge) in [9])

$RL-CW_{v_i}(seg_{ij})$: returns region label P_δ of the region encountered in the *clockwise* direction from seg_{ij} with viewing origin at v_i . (same as NFACE(edge) in [9])

In addition, we shall define two segments seg_{ij} and seg_{ik} to be *adjacent segments* if they share a common junction v_i and (1) $SegCCW_{v_i}(seg_{ij}) = seg_{ik}$ or (2) seg_{ik}

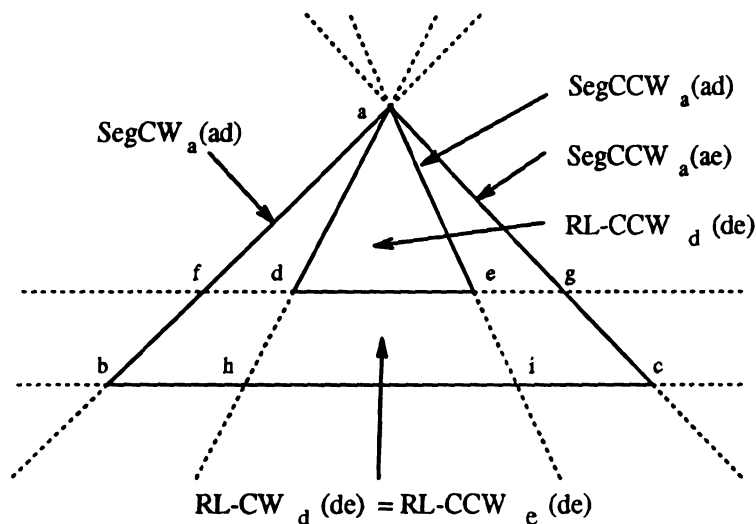


Figure 4.8. Graphical interpretation of newly defined functions.

$= SegCW_{v_i}(seg_{ij})$ and define the region *immediately bounded* by seg_{ij} and seg_{ik} as $RL-CCW_{v_i}(seg_{ij})$ (or equivalently $RL-CW_{v_i}(seg_{ik})$) if (1) is true or $RL-CW_{v_i}(seg_{ij})$ (or equivalently $RL-CCW_{v_i}(seg_{ik})$) if (2) is true.

The above function definitions are depicted graphically in Figure 4.8. Note that

$$RL-CCW_{v_i}(seg_{ij}) = RL-CW_{v_i}(seg_{ij}) \text{ and } RL-CCW_{v_j}(seg_{ij}) = RL-CW_{v_j}(seg_{ij}).$$

4.4.2 Geometric Constraints Imposed by Projection of a δ -polygon

Theorem 4.1 *If a circle centered at v is divided into n pieces by n straight segments, all extending from v to the boundary of the circle (like a pie chart), then the 2-coloring problem has a solution if and only if n is even.*

Pf: Assume $a_1, a_2, a_3, \dots, a_{n-1}, a_n$ are the pieces of the pie in the clockwise direction starting at a_1 . In a 2-coloring problem, adjacent sectors must have alternating colors. Therefore, one must color the pieces in the following way. If a_1 is colored green, a_2

must be colored white, a_3 must be colored green, and so on. When all the pieces are colored, a_i will be colored green for all odd i 's and a_j will be colored white for all even j 's.

Since a_1 and a_n are adjacent, they must also have different colors. If n is even, then a_1 and a_n will have different color and the 2-coloring problem is solved. However, if n is odd, both a_1 and a_n will have the same color, namely green. Since a_1 and a_n are adjacent to each other, the condition of 2-coloring is violated. Therefore, the 2-coloring problem has a solution if and only if n is even. \square

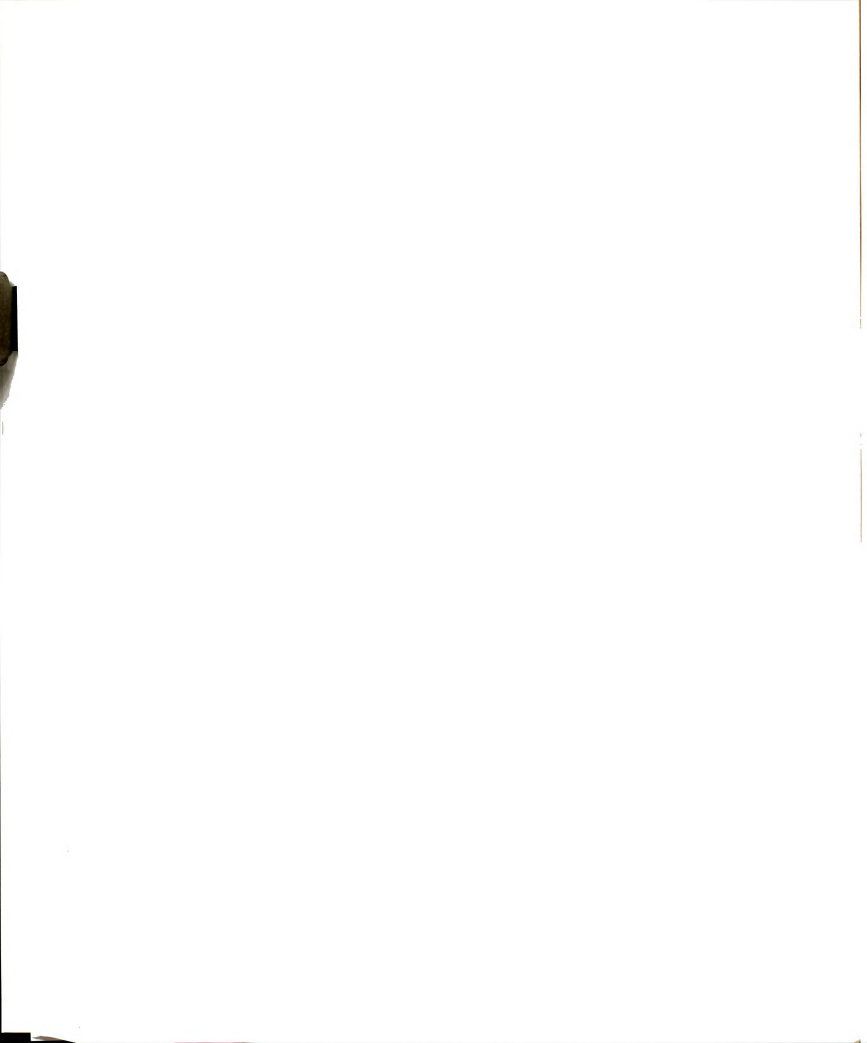
Theorem 4.2 *Every vertex of a δ -polygon must be incident with an even number of edges.*

Pf: Let v be a vertex of the δ -polygon, P_δ . Consider the plane that is coplanar with P_δ , call the polygonal plane and a circular neighborhood N_r of radius $r < \delta$, centered at v on this polygonal plane, the edges of P_δ incident with v divides N_r into sectors where each sector is wholly inside or outside of P_δ on the polygonal plane (recall that the definition of δ -polygon guarantees that each edge of P_δ have length greater than δ on the polygonal plane).

Suppose the vertex v is incident with odd, say $2n-1$, number of edges. Those $2n-1$ edges divides N_r into $2n-1$ sectors. Since there is an odd number of sectors, from Theorem 1, at least 2 adjacent sectors will have the same label. But this means that the edge between those two sectors will have the same label on both sides, which contradicts the definition of a δ -polygonal edge. Therefore, a δ -polygon may not have a vertex of odd degree. \square

Corollary 4.2.1 *Every junction of the line drawing of a non-accidental view of a δ -polygon must be incident with an even number of line segments.*

Pf: Recalling the definition of a non-accidental viewpoint (Section 2.3), the projection of a δ -polygon onto the image plane from a non-accidental viewpoint is an ϵ -polygon.



Therefore, by Theorem 4.2, every junction (vertex) of the line drawing (ϵ -polygon) must be incident with an even number of line segments (edges). \square

Theorem 4.3 *Let \mathbf{S} denote a set of coplanar (co-quadric) faces of objects in the scene, \mathbf{G} denote the perfect LDG of those faces in \mathbf{S} as viewed through some non-accidental viewpoint, and P_δ denote the region label of all the regions project onto which those faces of \mathbf{S} (P_δ is simply the equation of the coplanar (co-quadric) surface). If seg_{ij} and seg_{ik} are two adjacent segments in \mathbf{G} then either*

$$RL-CW_{v_i}(seg_{ij}) = P_\delta = RL-CCW_{v_i}(seg_{ik})$$

or

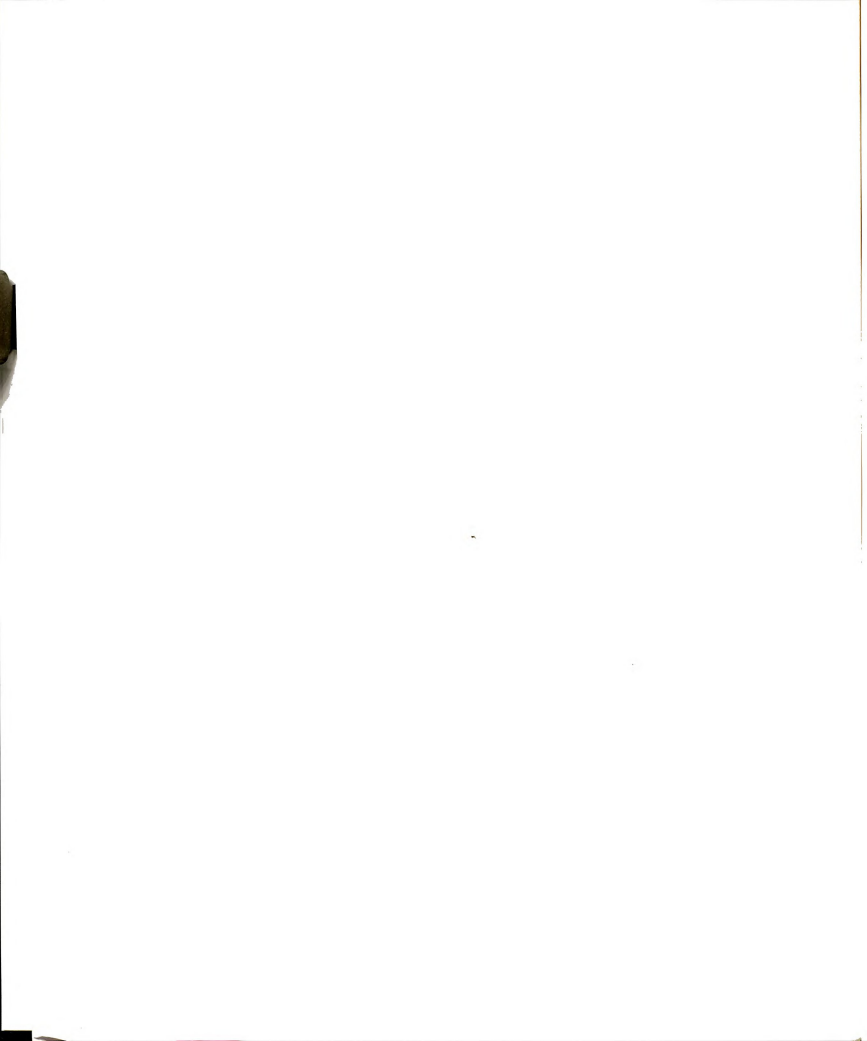
$$RL-CCW_{v_i}(seg_{ij}) = P_\delta = RL-CW_{v_i}(seg_{ik})$$

Pf: From the definition and the properties of the LDG, two regions adjacent to a line segment in the LDG correspond to projections from two faces in distinct surfaces; thus must have different region labels. Therefore, it must be that $RL-CCW_{v_i}(seg_{ij}) \neq RL-CW_{v_i}(seg_{ij})$ and $RL-CCW_{v_i}(seg_{ik}) \neq RL-CW_{v_i}(seg_{ik})$. Without loss of generality, label the regions of \mathbf{G} which are not images of faces in \mathbf{S} as P_x , where $P_x \neq P_\delta$.

First let's assume that the region immediately bounded by seg_{ij} and seg_{ik} corresponds to $RL-CCW_{v_i}(seg_{ij})$ and $RL-CW_{v_i}(seg_{ik})$ (see Figure 4.9 (a)). Thus $RL-CCW_{v_i}(seg_{ij}) = RL-CW_{v_i}(seg_{ik})$.

Case 1: Assume $RL-CCW_{v_i}(seg_{ij}) = P_\delta$. Then $RL-CCW_{v_i}(seg_{ij}) = P_\delta = RL-CW_{v_i}(seg_{ik})$.

Case 2: Assume $RL-CW_{v_i}(seg_{ij}) = P_\delta$. If it were the case that $RL-CW_{v_i}(seg_{ik}) = P_\delta$, then so must $RL-CCW_{v_i}(seg_{ij}) = P_\delta$ because they bound and label the same region in \mathbf{G} . But this implies that $RL-CCW_{v_i}(seg_{ij}) = RL-CW_{v_i}(seg_{ij}) = P_\delta$, which contradicts the fact that adjacent regions of a line segment cannot have the same region label. Thus, it must be that $RL-CCW_{v_i}(seg_{ik}) \equiv RL-CW_{v_i}(seg_{ij}) = P_\delta$.



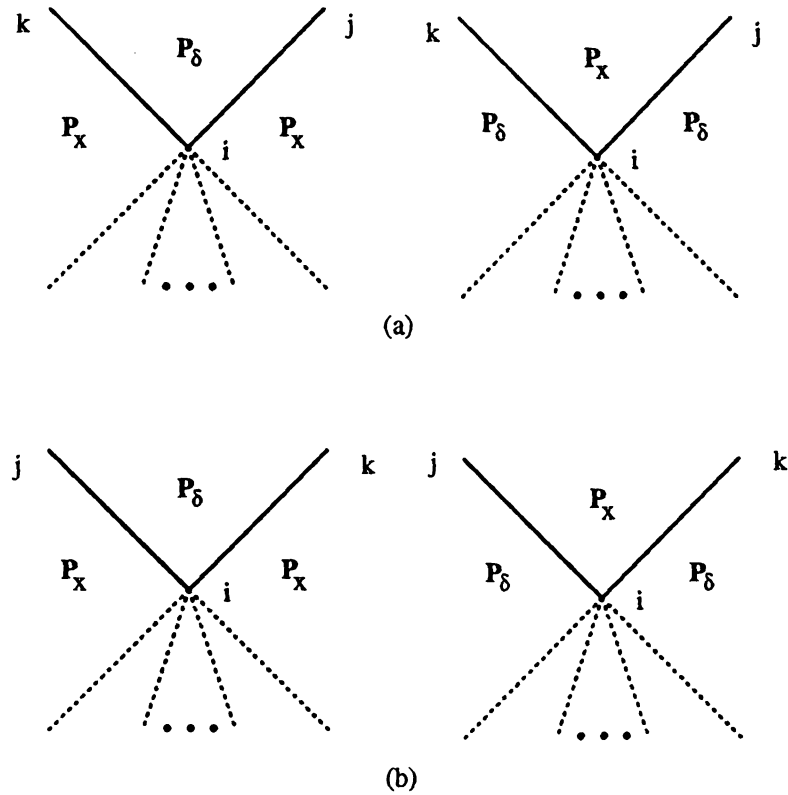


Figure 4.9. Graphical illustration of Theorem 4.3.

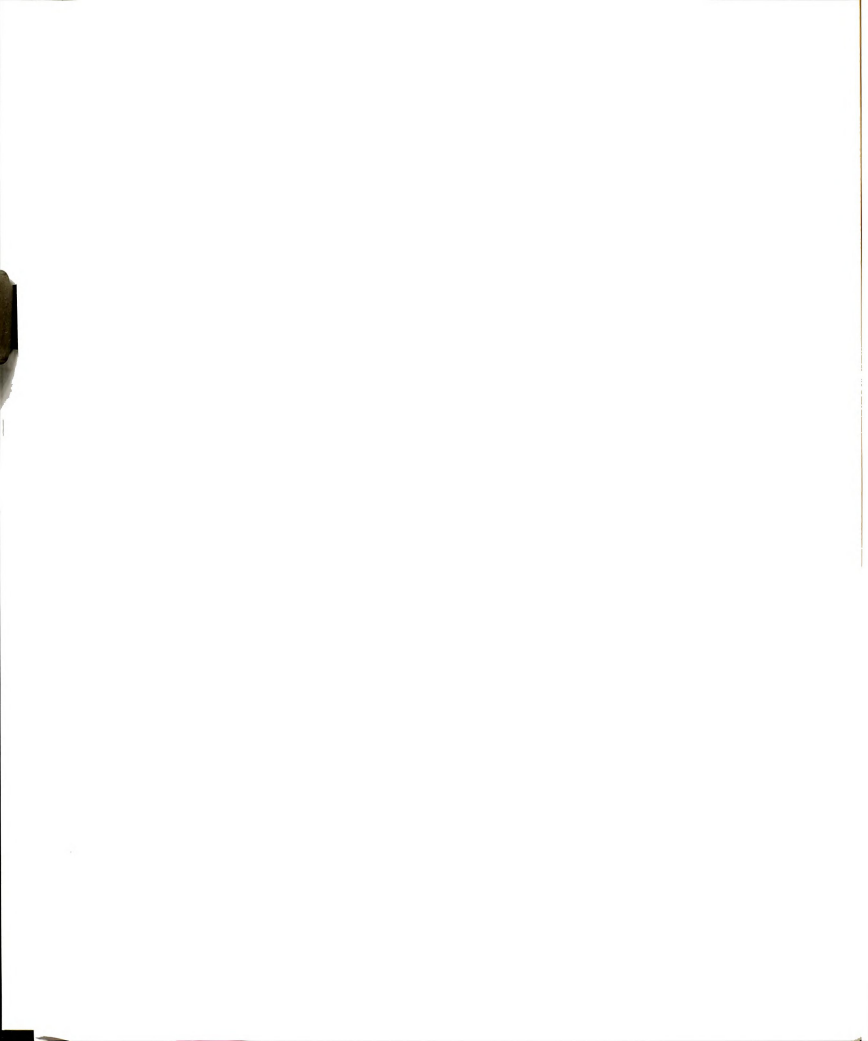
The same argument can be constructed if we assume the region immediately bounded by seg_{ij} and seg_{ik} corresponds to $RL-CW_{v_i}(seg_{ij})$ and $RL-CCW_{v_i}(seg_{ik})$ instead (see Figure 4.9 (b)). \square

Corollary 4.3.1 *Let S denote a set of coplanar (co-quadric) faces of objects in the scene, G denote the perfect LDG of those faces in S as viewed through some non-accidental viewpoint, and P_δ denote the region label of all the regions projected onto by those faces in S (P_δ is simply the equation of the surface). If seg_{ij} and seg_{ik} are two adjacent segments in G then either*

$$RL-CCW_{v_i}(seg_{ij}) = RL-CCW_{v_i}(seg_{ik})$$

nor

$$RL-CW_{v_i}(seg_{ij}) = RL-CW_{v_i}(seg_{ik}).$$



Pf: Since seg_{ij} and seg_{ik} are adjacent line segments, either $RL\text{-}CCW_{v_i}(seg_{ij}) = RL\text{-}CW_{v_i}(seg_{ik})$ or $RL\text{-}CW_{v_i}(seg_{ij}) = RL\text{-}CCW_{v_i}(seg_{ik})$ must hold. If $RL\text{-}CCW_{v_i}(seg_{ij}) = RL\text{-}CCW_{v_i}(seg_{ik})$ then by transitivity, $RL\text{-}CCW_{v_i}(seg_{ij}) = RL\text{-}CW_{v_i}(seg_{ij})$, which violates the fact that adjacent regions cannot have the same region label. Likewise, if $RL\text{-}CW_{v_i}(seg_{ij}) = RL\text{-}CW_{v_i}(seg_{ik})$ then $RL\text{-}CCW_{v_i}(seg_{ik}) = RL\text{-}CW_{v_i}(seg_{ik})$, which again is impossible. \square

4.4.3 Geometric Constraint Based Decision Rules

Before characterizing the decision rules, we must first define more terminology and redefine the functions given in the previous section.

Since the functions $RL\text{-}CW$ and $RL\text{-}CCW$ are defined as functions that act on the perfect LDG, they need to be redefined to be used here.

Let \mathbf{W} denote the set of coplanar wings on the plane P_δ that is being reconstructed and let \mathbf{G} denote the state of reconstruction of P_δ at a given time instance. Recall that each wing sample has information about the two surfaces that project to the two regions adjacent to the wing edge projection (called the wing segment) in the LDG. One of the regions adjacent to the wing segment must be labeled P_δ because wing samples in \mathbf{W} are all coplanar with plane P_δ and the other region must not be labeled P_δ , say P_x . When the wing segments are extended and intersected to form undecided-segments, the region labels on both sides of the wing segments are carried over to their extended segments. If wing segments happen to be collinear, then the extended segment(s) in-between the two wing segments may have multiple region labels on each side of the segment(s). Thus we shall denote the region label on each side of a segment in the LDG as a set of labels. When an undecided-segment becomes a must-segment, the region label on one side will be exactly P_δ and the other may not contain P_δ as a possible region label. An example is depicted in Figure 4.10.

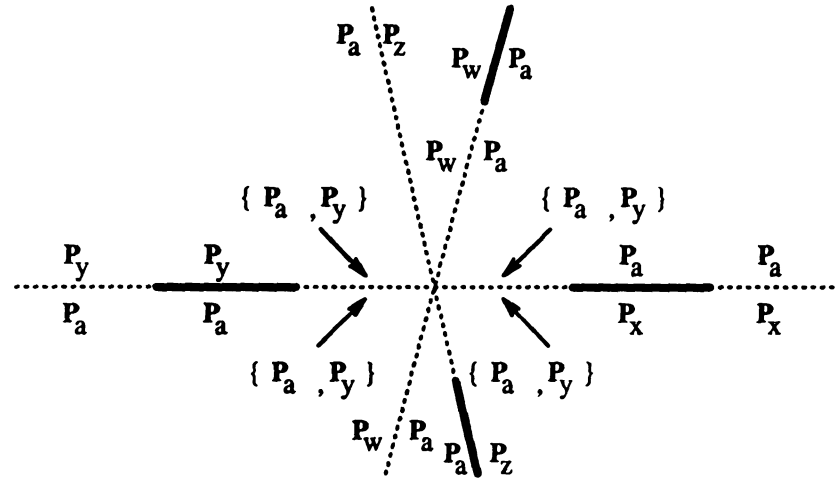


Figure 4.10. Multiple region labels are possible if wings are collinear.

We will now redefine the functions $RL-CCW$ and $RL-CW$ to suit the possibility of multiple region labels and to define three more new functions. These are the functions used in the implementation of the reconstruction algorithm.

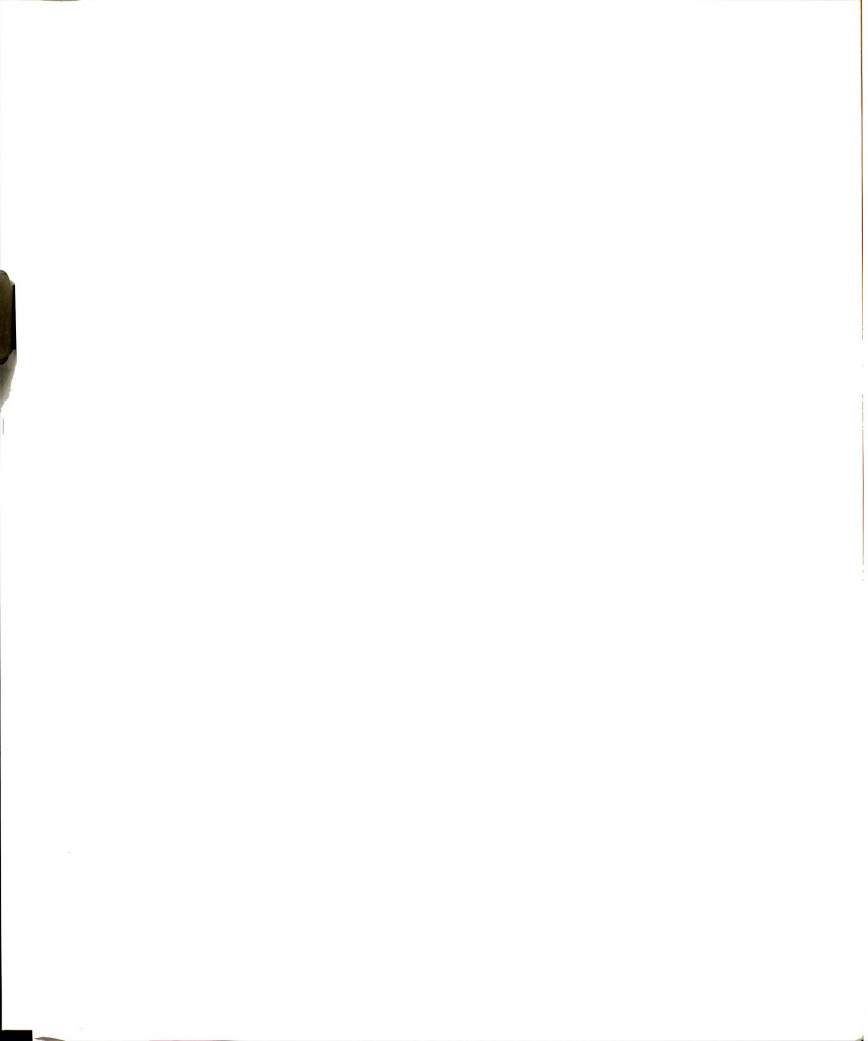
$RL-CCW_{v_i}(seg_{ij})$: returns the set of region labels associated with seg_{ij} on the side encountered in the *counter-clockwise* direction from seg_{ij} with viewing origin at v_i .

$RL-CW_{v_i}(seg_{ij})$: returns the set of region labels associated with seg_{ij} on the side encountered in the *clockwise* direction from seg_{ij} with viewing origin at v_i .

$WingDir_{v_i}(seg_{ij})$: returns True if at least one of the detected wings is collinear with seg_{ij} and appears on the v_j side of seg_{ij} using v_i as viewing origin; else returns False.

$deg_{must}(v_i)$: returns the number of must-segments currently known to be incident with junction v_i . Also called the *must-degree* of v_i .

$deg_{und}(v_i)$: returns the number of undecided-segments currently known to be incident with junction v_i . Also called the *undecided-degree* of v_i .



In addition to the list of must-, undecided- and unwanted-segments, three sets of junctions, V_{E_1} , V_{O_1} and V_{und} are also maintained for the new decision procedure. Members of the same set share the common characteristics as defined below.

$$V_{E_1} = \{ v_i \mid \deg_{must}(v_i) \text{ is even and } \deg_{und}(v_i) = 1. \}$$

$$V_{O_1} = \{ v_i \mid \deg_{must}(v_i) \text{ is odd and } \deg_{und}(v_i) = 1. \}$$

$$V_{und} = \{ v_i \mid \deg_{und}(v_i) \geq 1. \}$$

Note that $V_{E_1} \subseteq V_{und}$, $V_{O_1} \subseteq V_{und}$ and $V_{E_1} \cap V_{O_1} = null$.

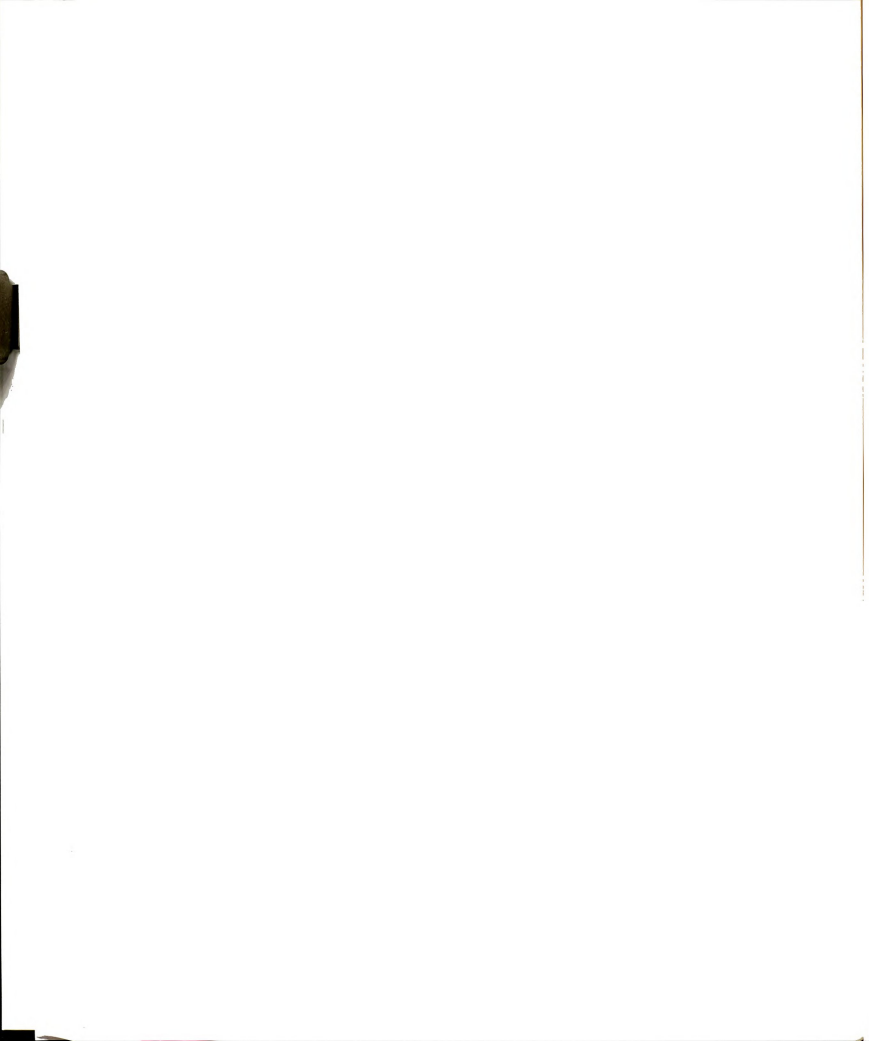
We make a note that the set membership of V_{E_1} , V_{O_1} and V_{und} and the values returned by the functions $SegCCW$, $SegCW$, $RL-CCW$, $RL-CW$, \deg_{must} , \deg_{und} may vary over time as the state of reconstruction G progresses. Therefore, the functions are not true functions in the mathematical sense but are access functions in the abstract data type sense. Those functions and sets are graphically illustrated in Figure 4.11.

As a direct consequence of Corollary 4.2.1, if a junction in G has exactly one undecided-segment, then that undecided segment can be classified as a must-segment or unwanted-segment based solely on the must-degree of that junction. This gives rise to our first two rules.

Rule 4.1 *An undecided-segment seg_{ij} can be discarded if $\deg_{must}(v_i)$ is even and $\deg_{und}(v_i) = 1$; or if $\deg_{must}(v_j)$ is even and $\deg_{und}(v_j) = 1$.*

Rule 4.2 *An undecided-segment seg_{ij} is a must-segment if $\deg_{must}(v_i)$ is odd and $\deg_{und}(v_i) = 1$; or if $\deg_{must}(v_j)$ is odd and $\deg_{und}(v_j) = 1$.*

From our experiments, most of the undecided-segments can be resolved using those 2 rules. However, there are situations where these two rules do not apply. We derived Rules 3a and 3b to discard more undecided-segments that cannot possibly be



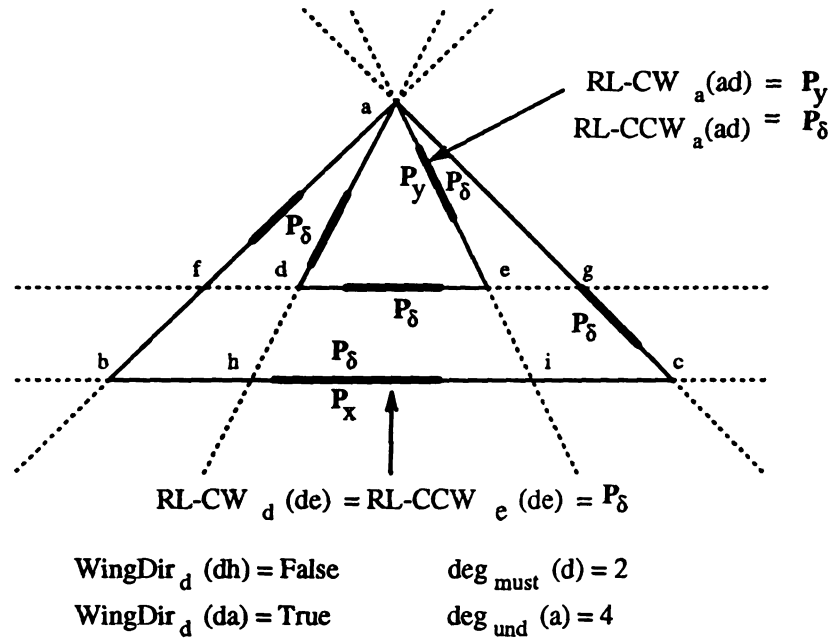


Figure 4.11. Graphical illustration of redefined and new functions. Notation: broken lines are undecided-segments, solid lines are must-segments, thick solid lines are wing segments.

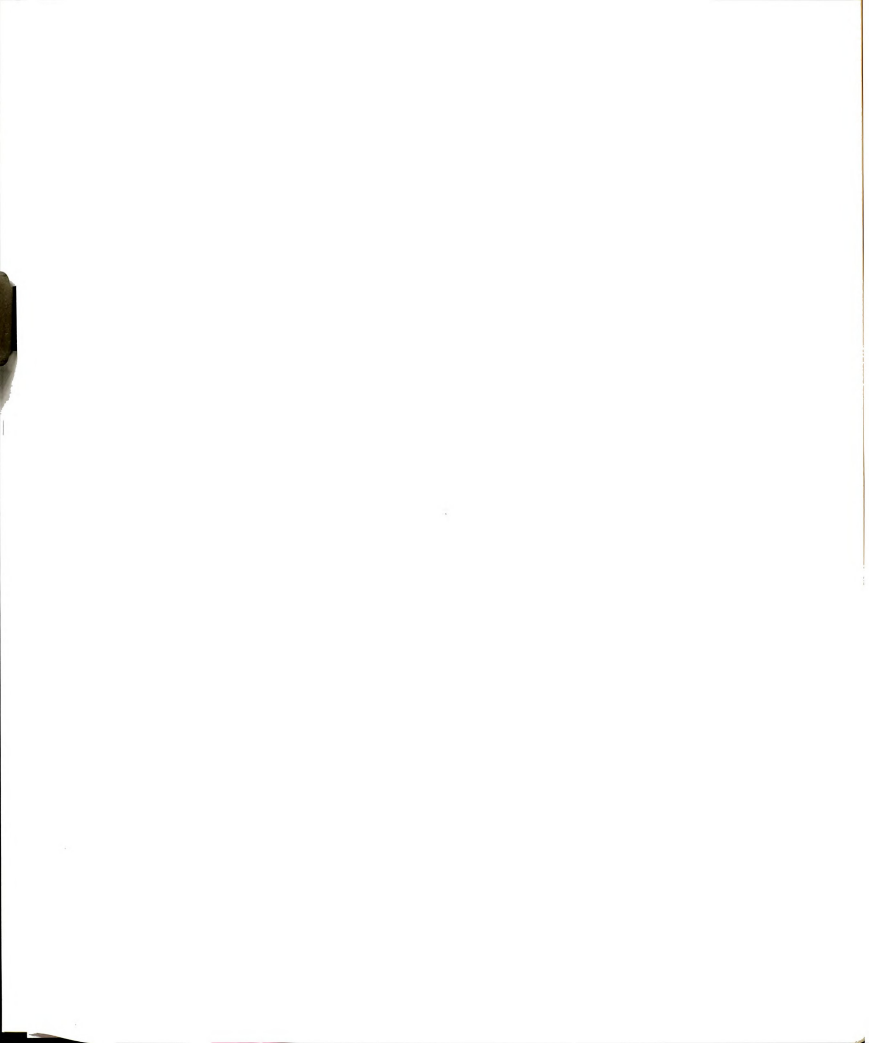
must-segments in the presence of other must-segments. These two rules are directly implied by Theorem 4.3 and Corollary 4.3.1.

Rule 4.3a *Given two adjacent line segments $Mseg_{ij}$ and $Useg_{il}$ where $Mseg_{ij}$ is a must-segment and $Useg_{il} = SegCCW_{v_i}(Mseg_{ij})$ is an undecided-segment. Assume that the wing group being processed has the surface representation P_δ . Then $Useg_{il}$ can be discarded as an unwanted-segment if*

$$RL-CCW_{v_i}(Mseg_{ij}) = \{P_\delta\} \text{ and } P_\delta \notin RL-CW_{v_i}(Useg_{il})$$

or

$$RL-CCW_{v_i}(Mseg_{ij}) \neq \{P_\delta\} \text{ and } RL-CW_{v_i}(Useg_{il}) = \{P_\delta\}.$$



Rule 4.3b *Given two adjacent line segments $Mseg_{ij}$ and $Useg_{il}$ where $Mseg_{ij}$ is a must-segment and $Useg_{il} = SegCW_{v_i}(Mseg_{ij})$ is an undecided-segment. Assume that the wing group being processed has the surface representation P_δ . Then $Useg_{il}$ can be discarded as an unwanted-segment if*

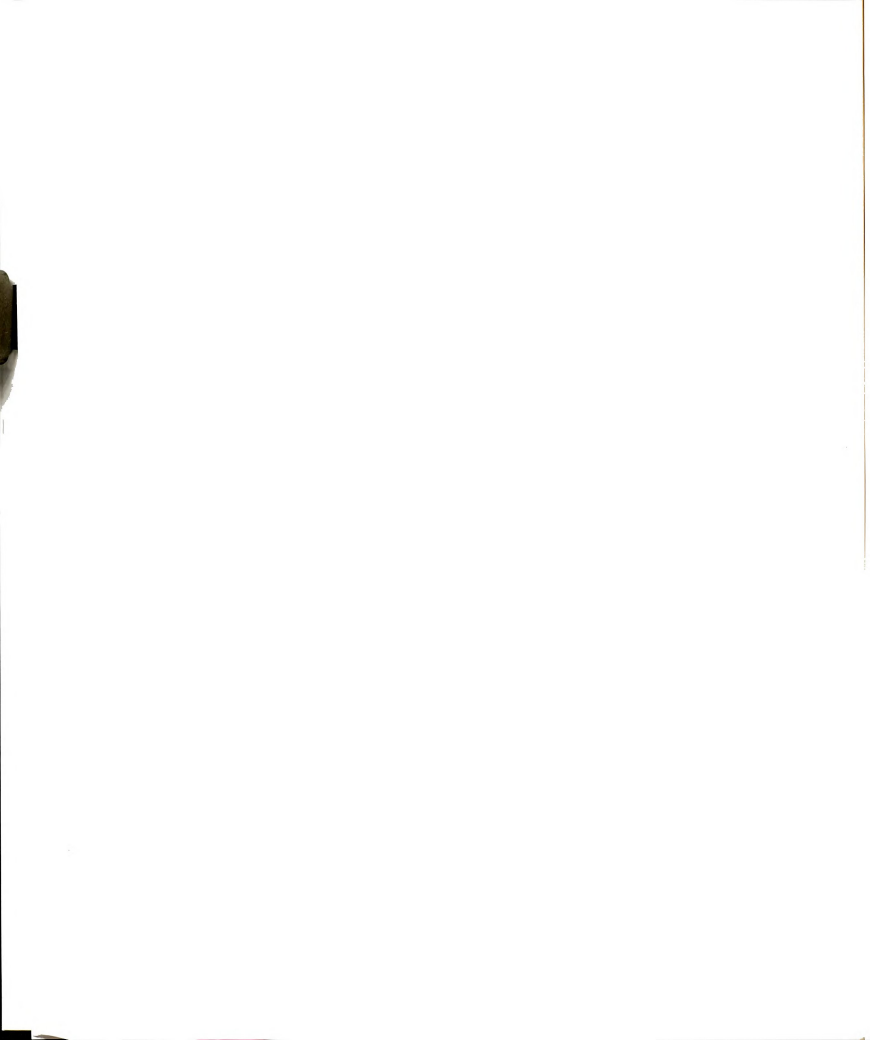
$$RL-CW_{v_i}(Mseg_{ij}) = \{P_\delta\} \text{ and } P_\delta \notin RL-CCW_{v_i}(Useg_{il})$$

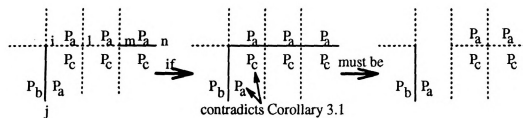
or

$$RL-CW_{v_i}(Mseg_{ij}) \neq \{P_\delta\} \text{ and } RL-CCW_{v_i}(Useg_{il}) = \{P_\delta\}.$$

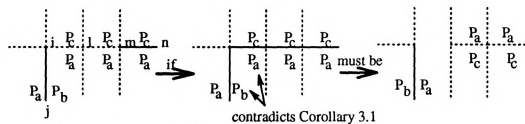
The situations depicted by **Rules 4.3 (a) and 3b** are graphically illustrated in Figure 4.12. Basically, **Rule 4.3** discards an undecided-segment if (1) it adjoins and neighbors a must-segment and (2) only the must-segment or the undecided-segment labels the commonly shared region as P_δ but not both. Figures 4.12(a)-(b) depict two situations in which the undecided-segment (broken segment) is discarded and Figures 4.12(c)-(d) show instances in which the rule would not apply.

Finally, we add a rule to remove spurious extension of a wing. Assumption A_3 states that at least one wing is detected on each visible edge of the scene. Thus, all extensions of a wing beyond intersection with a must-segment should all be removed. This is graphically demonstrated in Figure 4.13. In Figure 4.13(a), undecided-segments \overline{ij} and \overline{jk} are extensions of wing w_1 which forms the line l . Since must-segment m_2 intersects line l , if any of the segments on the \overline{ij} side of line l were to be a must-segment then there must be a wing on that side of line l . However, since there were no wings detected on that side, and there are no missing wings (Assumption A_3), all segments on the \overline{ij} side of line l must be unwanted-segments.

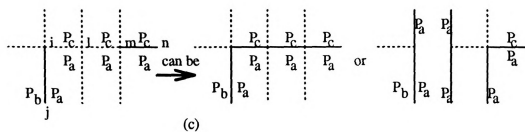




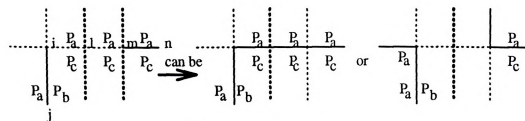
(a)



(b)

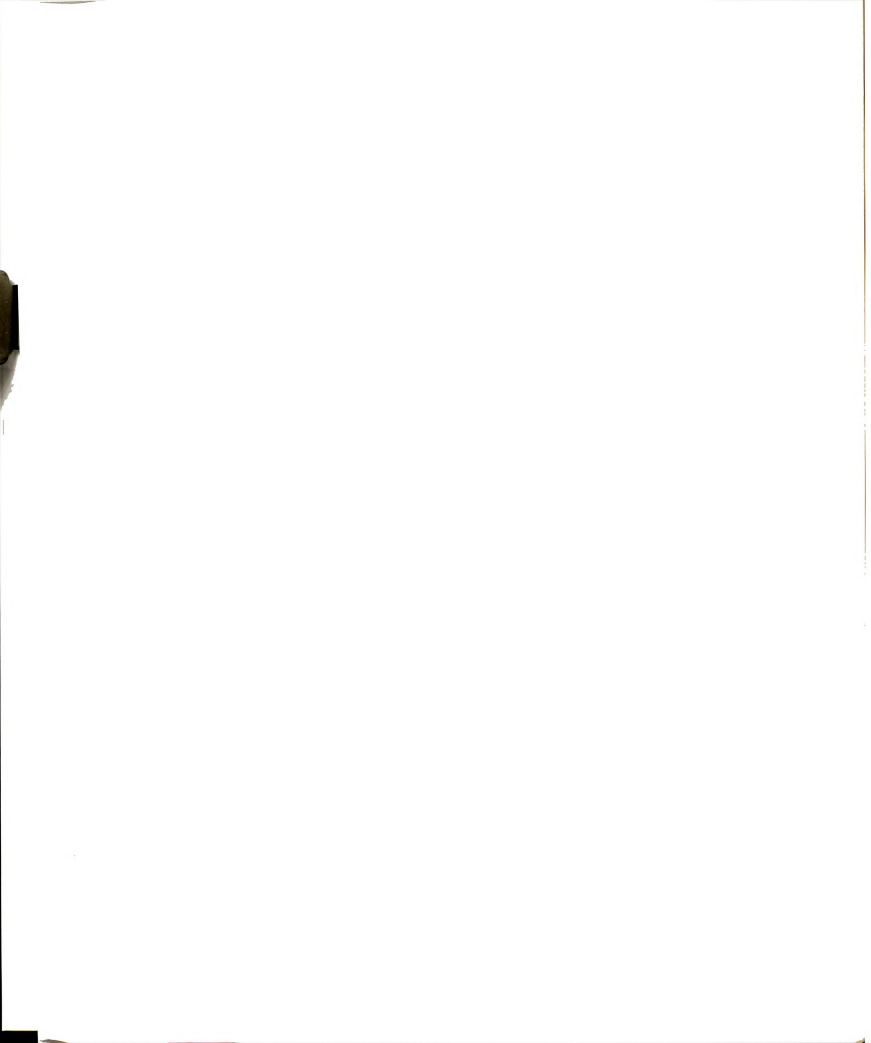


(c)



(d)

Figure 4.12. Graphical illustration of **Rule 4.3**. Must-segments and undecided-segments are represented by solid and broken lines, respectively. (a) Undecided segment $\overline{1}$ is discarded via **Rule 4.3a**. (b) Undecided segment $\overline{1}$ is discarded via **Rule 4.3b**. (c)-(d) Neither **Rule 4.3a** nor **Rule 4.3b** applies.



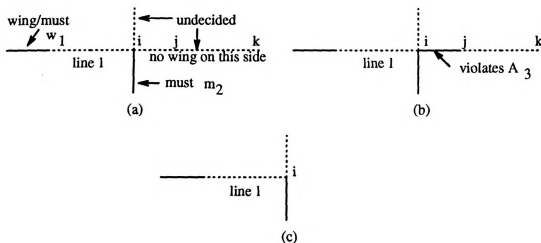


Figure 4.13. Graphical illustration of **Rule 4.4**. Must-segments and undecided-segments are represented by solid and broken lines, respectively. (a) No wing on the \overline{ij} side of line l . (b) if \overline{ij} were a must segment, it would violate Assumption A_3 . (c) Segments on \overline{ij} side of line l in (b) are unwanted-segments.

Rule 4.4 Given a must-segment $Mseg_{ij}$ and an undecided-segment $Useg_{ik}$ where $Useg_{ik} = SegCCW_{v_i}(Mseg_{ij})$ or $SegCCW_{v_i}(Mseg_{ij})$. If $Mseg_{ij}$ and $Useg_{ik}$ are not collinear (co-curvilinear) and $WingDir_{v_i}(Useg_{ik}) = False$, then all (undecided) segments collinear (co-curvilinear) to and on the side of $Useg_{ik}$ with respect to v_i including $Useg_{ik}$ can be deleted.

4.4.4 Complete Reconstruction Algorithm: POLY-2

We are now ready to give the geometric constraint based LDG reconstruction algorithm. All steps but step (2.4) of this algorithm are the same as the previous algorithm. Note that we apply **Rules 4.1** and **4.2** whenever possible because they are easier to apply than the other rules.

- (1) Cluster wing segments by plane equation
- (2) For each plane group
 - (2.1) generate a line for each wing
 - (2.2) compute the intersections among all lines
 - (2.3) preserve line segments which overlap a wing
 - (2.4) remove or preserve line segments by using Rules 4.1, 4.2, 4.3, and 4.4
 - (2.4.1) if $V_{E1} \neq null$, apply Rule 4.1 then goto (2.4.1)
 - (2.4.2) if $V_{O1} \neq null$, apply Rule 4.2 then goto (2.4.1)
 - (2.4.2) apply Rules 4.3 and 4.4 to a junction v_i of V_{und}
 - (2.4.3) if $V_{und} = null$ or no more changes are possible then goto (2.5)
else goto (2.4.1)
 - (2.5) delete redundant intersections
- (3) Merge LDGs of individual polygonal faces
- (4) Compute 3D vertices by intersecting line equations
- (5) Retain all line segment labels from wings
- (6) Output LDG with plane equations, 3D vertices and edges

4.4.5 Example - Paddle Wheel

Figure 4.14 depicts how one group of wings can be reconstructed using the above 4 rules. This group of wings are extracted from the "background" group (denoted as P_δ) of the paddle wheel example in Figure 4.19 (a). After step (2.3), the state of reconstruction looks like Figure 4.14 (a). The three junction lists, V_{E1} , V_{O1} and V_{und} are formed at this instance. By repeated application of Rules 4.1 and 4.2, 20 of the 40 undecided-segments were discarded as unwanted-segments and 3 became must-segments. Note that in Figure 4.14 (b), junctions b_5 and $b_8 \in V_{und}$ due to accidental intersections, but they are eliminated soon after. The intermediate LDG resulting from each application of these two rules are shown in Figure 4.14 (b)-(f). At this point, both V_{E1} and V_{O1} are null lists, but there are still some undecided-segments in the LDG. Thus a more complex rule must be used to further the reconstruction

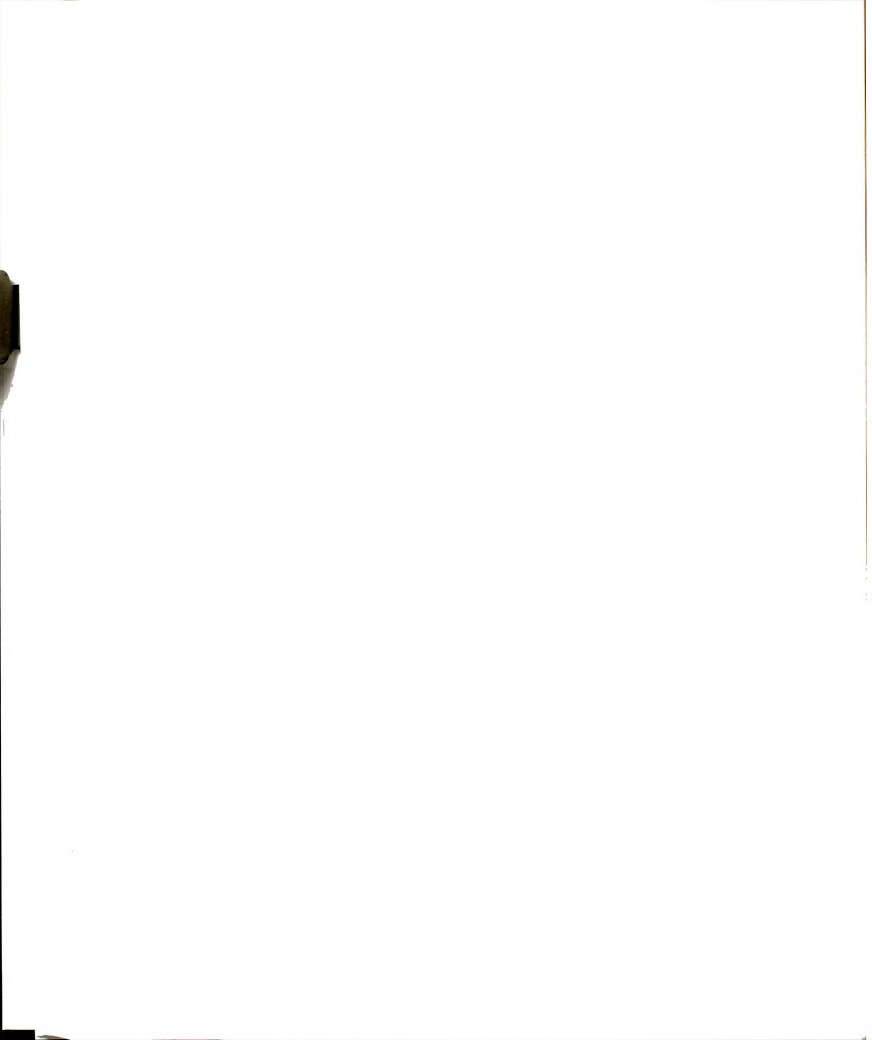
Table 4.1. Rest of the processing steps of the Paddle Wheel example.

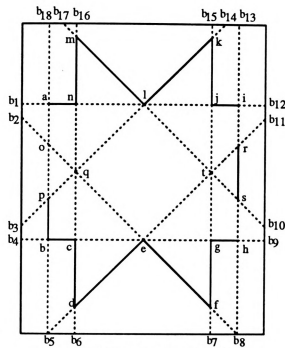
Iteration	V_{E_1}	V_{O_1}	V_{und}	Next Action
10	$\{o\}$	null	$\{c, e, g, j, l, o, q, t\}$	R 4.1 on $\{o\}$, removes nl
11	null	null	$\{c, e, g, j, l, q, t\}$	R 4.3 on $\{c\}$, removes $c\bar{q}$
12	$\{c, q\}$	null	$\{c, e, g, j, l, q, t\}$	R 4.1 on $\{c, q\}$, removes $c\bar{e}$ and $\bar{q}e$
13	null	null	$\{e, g, j, l, t\}$	R 4.4 on $\{e\}$, removes $\bar{e}t$
14	$\{e\}$	null	$\{e, g, j, l, t\}$	R 4.1 on $\{e\}$, removes $\bar{e}g$
15	$\{g\}$	null	$\{g, j, l, t\}$	R 4.1 on $\{g\}$, removes gt
16	null	null	$\{j, l, t\}$	R 4.3 on $\{l\}$, removes lt
17	$\{l, t\}$	null	$\{j, l, t\}$	R 4.1 on $\{l, t\}$, removes lj and jt
18	null	null	null	DONE

process. The undecided-segment $\bar{n}l$ and the must-segment $\bar{n}\bar{m}$ satisfy the criteria set forth in **Rule 4.3** (i.e., $RL-CW_n(Mseg_{nm}) \neq \{P_5\}$ and $RL-CCW_n(Useg_{nl}) = \{P_5\}$). Therefore, $\bar{n}l$ can be discarded. Upon removal of $\bar{n}l$ (see Figure 4.14(g)), **Rule 4.1** can be applied once more to arrive at Figure 4.14(h). This is followed by one application of **Rule 4.4** to remove undecided-segments $\bar{p}q$ and $\bar{q}l$. As a result, more undecided-segments can be discarded or become must segments via application of **Rules 4.1** and **4.2**. Table 4.1 shows the rest of the actions to be taken to arrive at the final reconstructed “background” LDG in Figure 4.15.

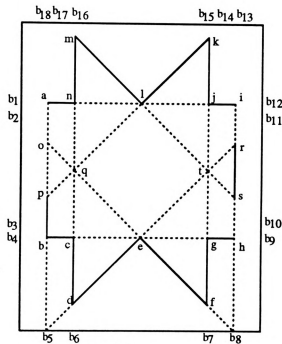
4.4.6 Merging of Surfaces

The surface merging step is trivial for the first algorithm since every visible surface in the scene is guaranteed to be completely recovered. However, in the second algorithm where surfaces are reconstructed using the four rules mentioned above, it is possible to have some undecided-segments still unresolved at the end of the planar surface reconstruction step. That is, those four rules only depict necessary but not sufficient conditions for complete and unique reconstruction of surfaces. An example is illustrated in Figure 4.16. Figure 4.16 (a) shows the wings detected for the front face of the block as in Figure 4.1. The reconstructed LDG using the above proce-

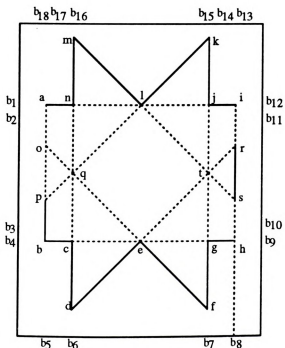




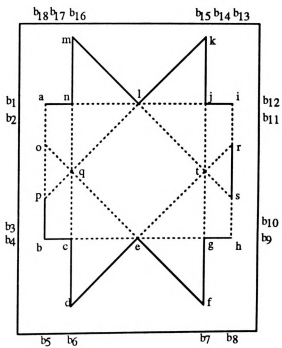
- $V_{E1} = \{ b1-b4, b6, b7, b9-b18 \}$
 (a) $V_{O1} = \text{empty}$
 $V_{und} = \text{all the junctions in the figure}$



- $V_{E1} = \{ b, d, f \}$
 (b) $V_{O1} = \{ a, i \}$
 $V_{und} = \{ b5, b8, a-j, l, n-t \}$

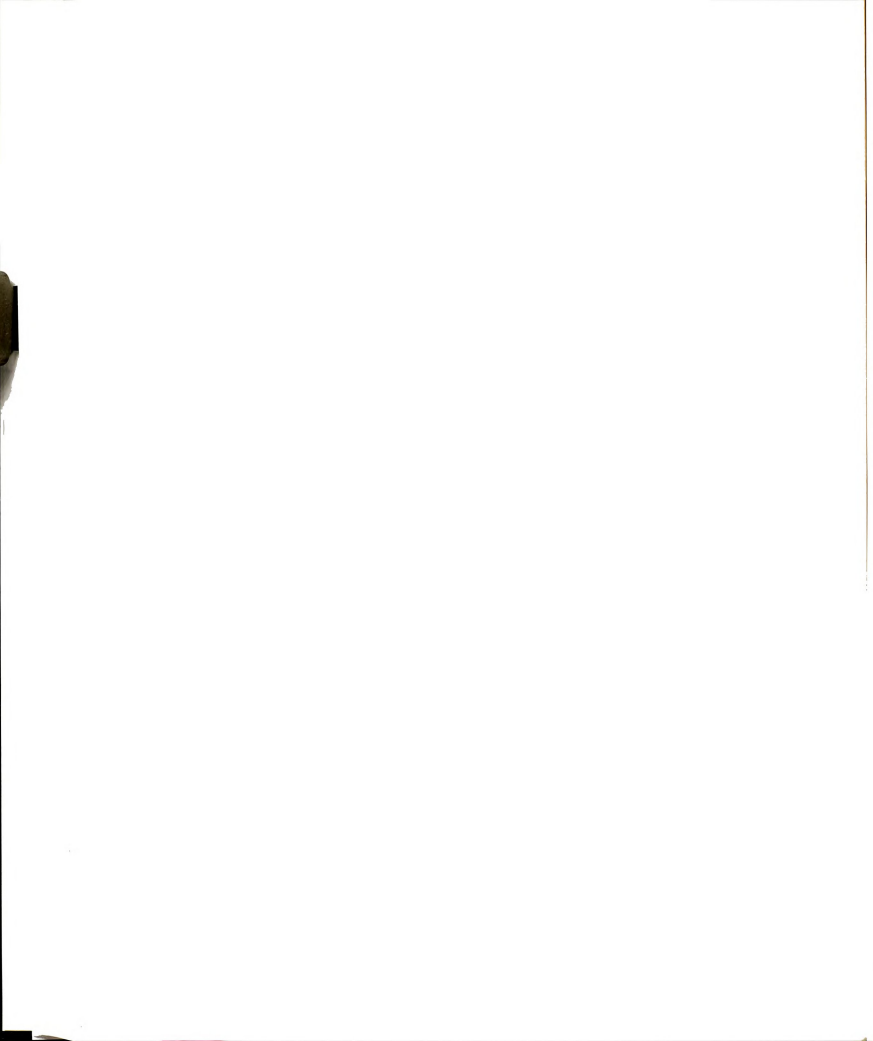


- $V_{E1} = \{ b8 \}$
 (c) $V_{O1} = \{ a, i \}$
 $V_{und} = \{ b8, a, c, e, g-j, l, n-t \}$



- $V_{E1} = \text{empty}$
 (d) $V_{O1} = \{ a, h, i \}$
 $V_{und} = \{ a, c, e, g-j, l, n-t \}$

Figure 4.14. cont. (see page 128 for caption)



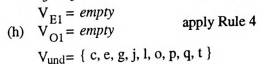
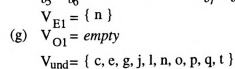
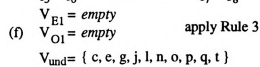
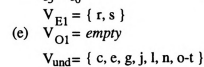
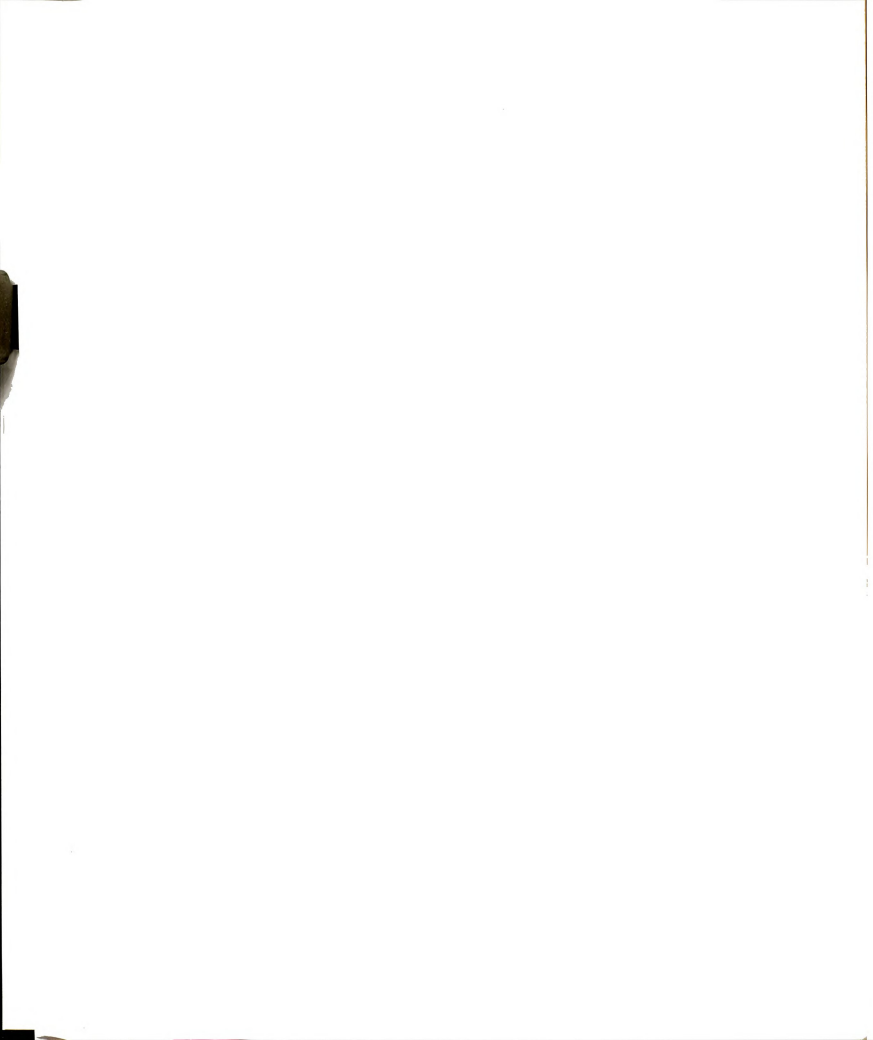


Figure 4.14. cont. (see page 128 for caption)



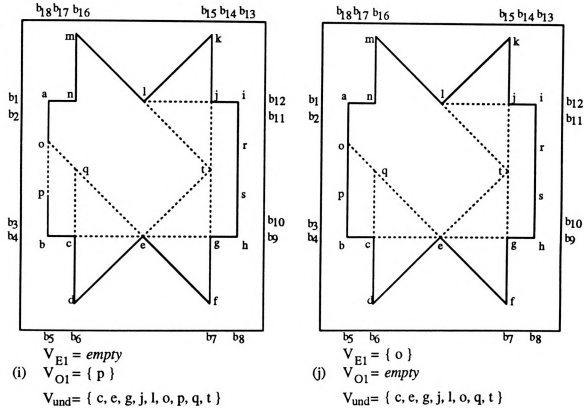
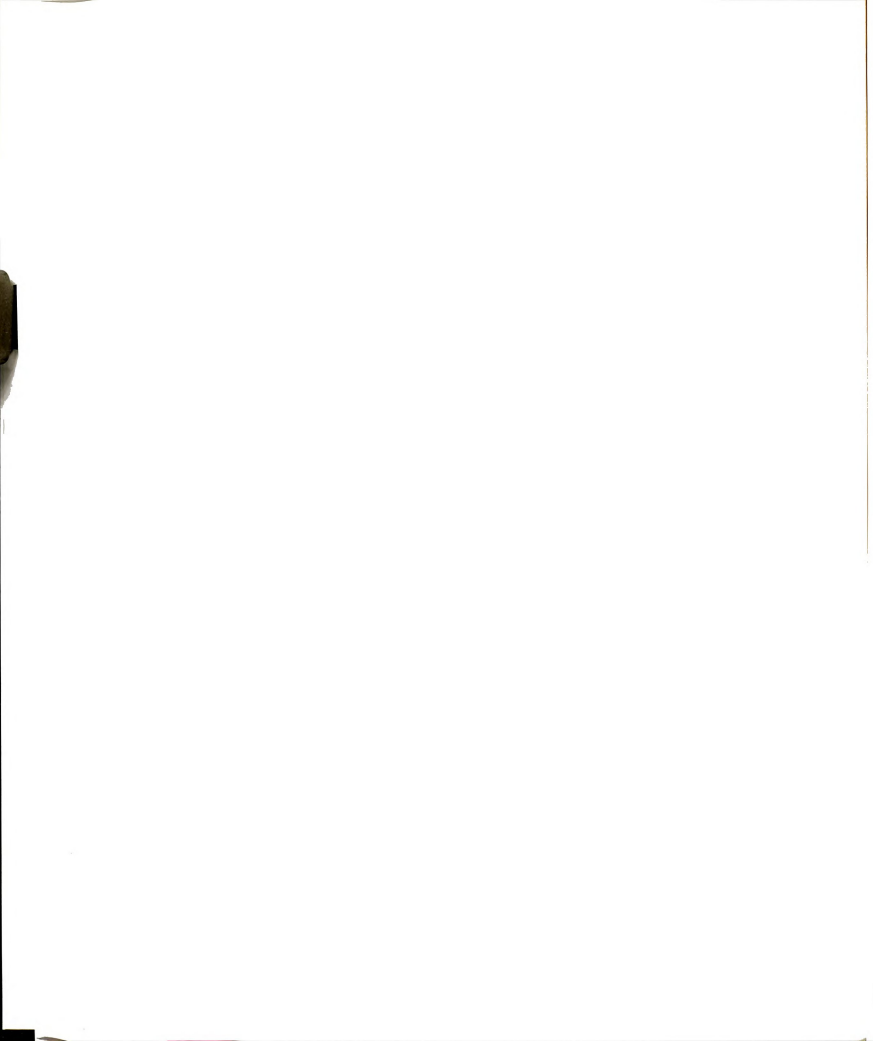


Figure 4.14. Reconstruction of the background wing group of the Paddle Wheel.

- (a) State of reconstruction after step (2.3).
- (b) Result of applying **Rule 4.1** to (a).
- (c) Result of applying **Rule 4.1** to (b).
- (d) Result of applying **Rule 4.1** to (c).
- (e) Result of applying **Rule 4.2** to (d).
- (f) Result of applying **Rule 4.1** to (e).
- (g) Result of applying **Rule 4.3** to segment \overline{nl} in (f).
- (h) Result of applying **Rule 4.1** to (g).
- (i) Result of applying **Rule 4.4** to segments \overline{pq} and \overline{ql} in (h).
- (j) Result of applying **Rule 4.2** to (i).



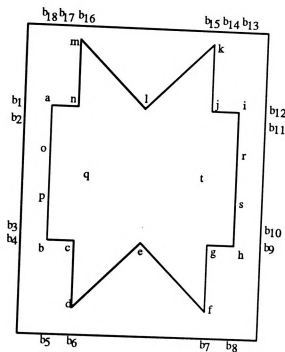
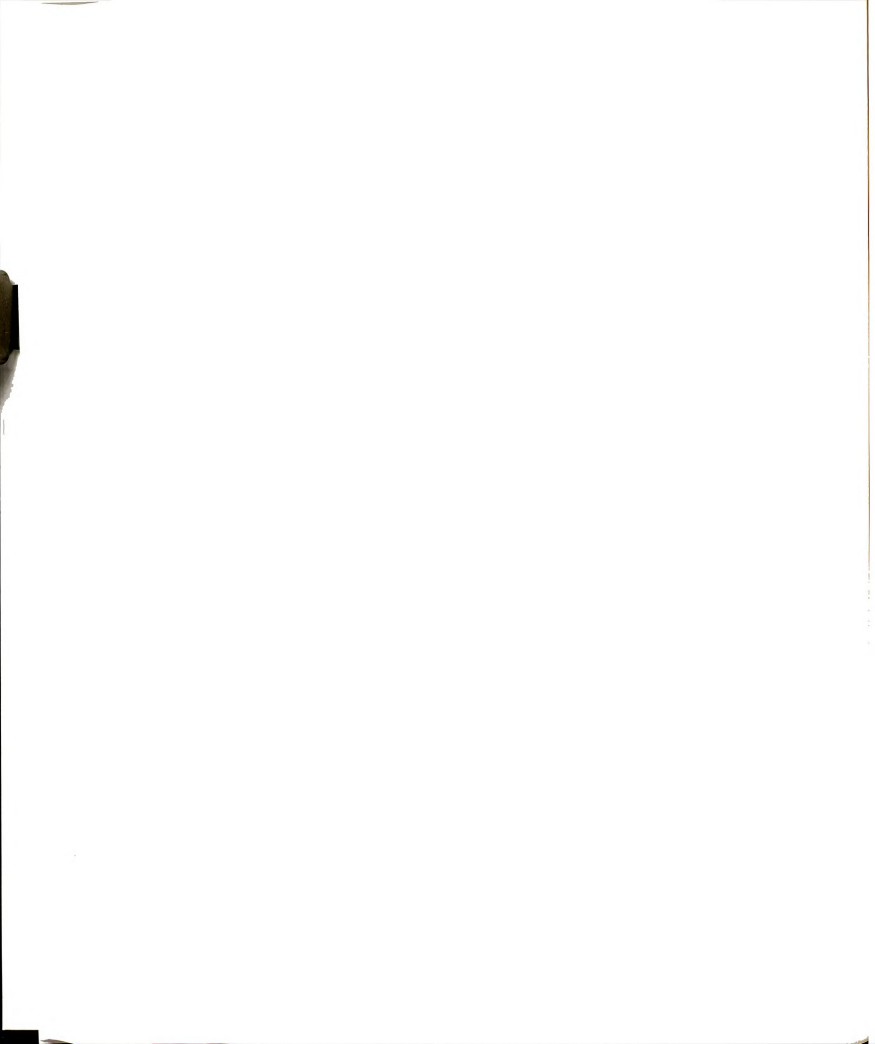


Figure 4.15. Reconstructed silhouette (background) of the Paddle Wheel.

ture results in eight undecided-segments that cannot be resolved using the 4 rules (see Figure 4.16 (b)). There are four possible valid interpretations as shown in Figures 4.16(c)-(f). Without information about other adjacent surfaces, this ambiguity cannot be resolved.

Although the case depicted in Figure 4.16 is highly coincidental, nevertheless it shows that the preceding algorithm cannot guarantee a unique LLDG even with a wing sample from every edge visible in the LDG. Even the use of a junction catalogue cannot resolve this ambiguity. However, we must note that since all four possible interpretations are physically realizable, perhaps it is not that the rules are insufficient, but rather that the wing representation of the scene is inherently ambiguous. Unless the wings are made to be as long as the actual edge segments, ambiguous cases can always be constructed. Unfortunately, this requirement would mean derivation of a perfect line drawing graph from the raw input image.



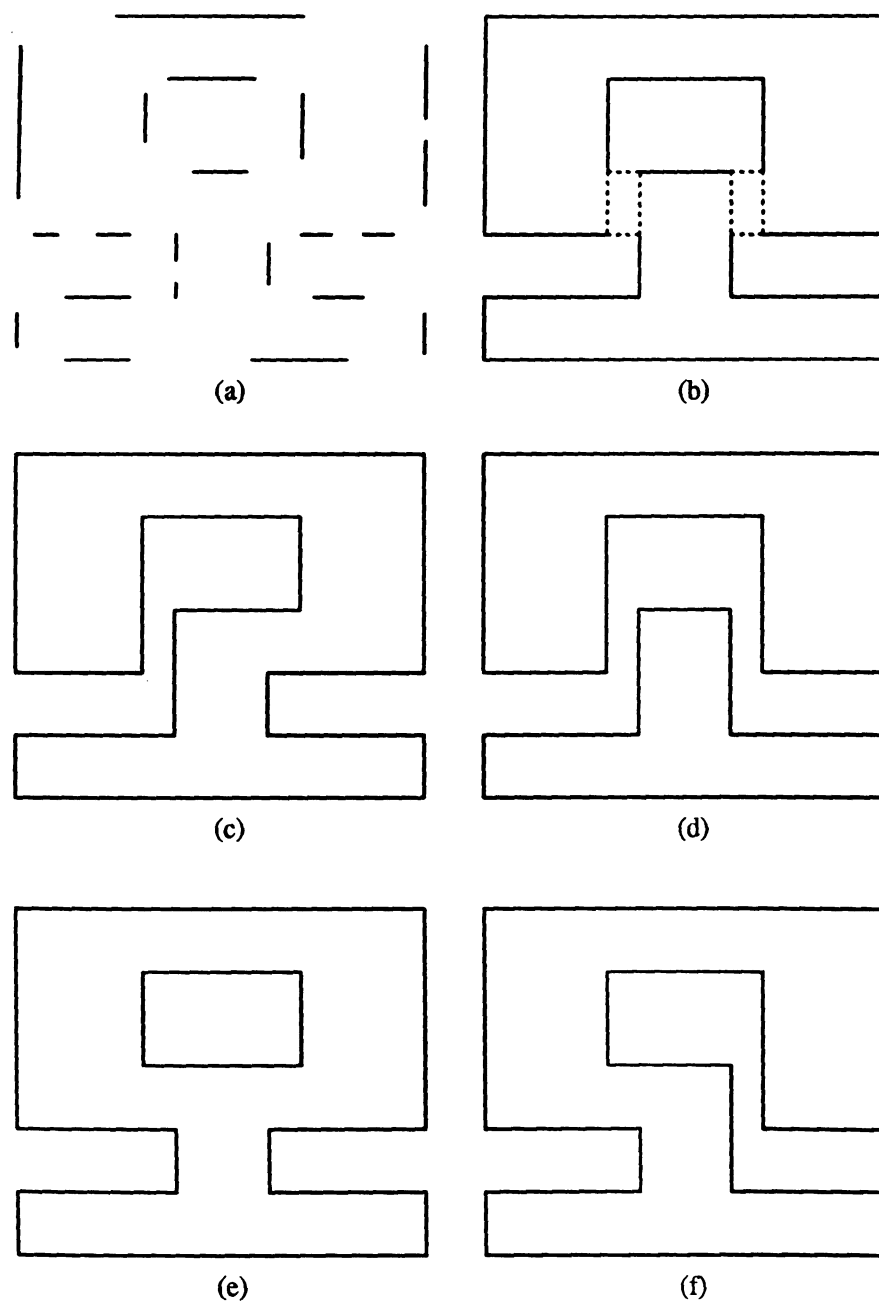


Figure 4.16. Algorithm 2 on front face of Big block. (a) Wings detected. (b) 4 undecided segments still unresolved from surface reconstruction. (c)-(f) Possible valid interpretations (only (e) remains after merging surfaces).

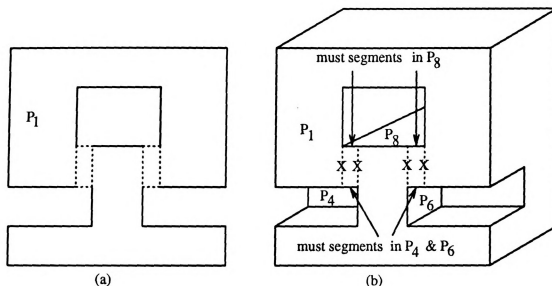


Figure 4.17. Reconstructed LDG of the Big Block. (a) Incomplete front face. (b) Ambiguity resolved when surfaces are merged.

With this algorithm, the fate of any left over undecided-segments on a surface is determined when the surface is merged with its neighboring surfaces, if any exist. Basically, each of the remaining undecided-segments will become a must-segment only if it is completely overlapped by a must-segment on some adjacent surface. In other words, since a segment can only be adjacent to exactly two surfaces, if the segment is deemed to be a must-segment on one surface, then it must also be a must-segment on the other surface. Using this fact, when all the surfaces of the 3D Big-Block are merged, the algorithm is able to resolve the ambiguous cases in Figure 4.16. The result is shown in Figure 4.17.

4.5 Experimental Results

These two line drawing graph reconstruction algorithms have been implemented in common LISP and tested on more than 15 test cases. The average run time for scenes considered in this section was less than 5 seconds per scene on a Sun690MP. Wing

detection time is not included in this estimate. Both syntactically generated wing sets and wings detected from real images were used to test the two algorithms. For the syntactically generated cases, the completely labeled line drawing graph is first created by hand. The wings are then extracted from each line in the line drawing. Note that each line segment in the graph can produce more than one wing. The results of the experiments on the syntactic data are depicted in Figures 4.18 and 4.19.

Figure 4.18 shows the input wing set and all the reconstructed faces as well as the final reconstructed LLDG of the Big Block example. Note that although face P_1 cannot be uniquely reconstructed using the geometric constraints based algorithm, the merging of all the faces discards the unsupported cases and results in unique interpretation of the object in the scene. More examples showing only the input wings and the reconstructed LLDG are given in Figure 4.19.

The geometric constraint based algorithm was also applied to a set of wings detected from a real image using the wing detector outlined in Chapter 3. Since the reconstruction algorithm demands complete wing samples (at least one wing on every edge contour), we have doctored the output of the wing detector by inserting one missing wing and merged collinear compatible wings into one long wing. The augmented wing set was then fed into the algorithm for reconstruction. Results of the reconstruction is shown in Figure 4.20.

4.6 Worst Case Time Complexities of POLY-1 and POLY-2

Both POLY-1 and POLY-2 algorithms consist of four sequential modules: clustering of wings, extending wings and computing the intersections, resolving the undecided-segments and merging of the planar faces. Analysis of the complexity of the algorithms

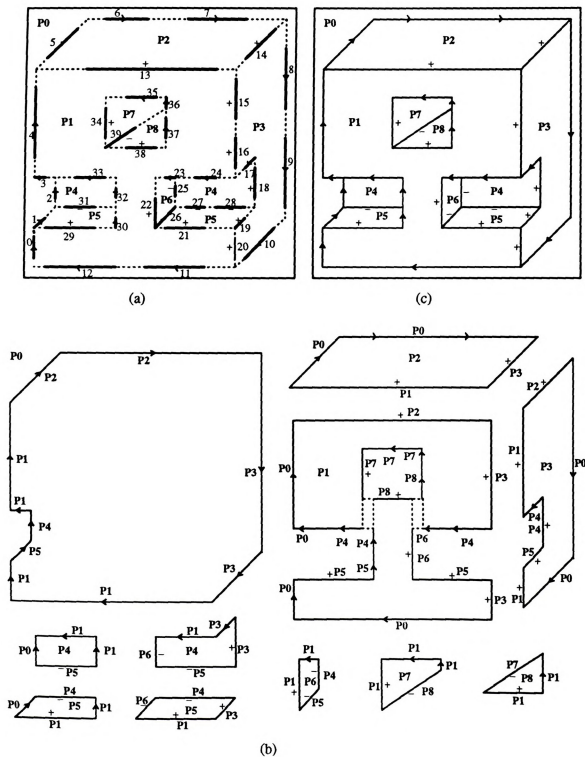
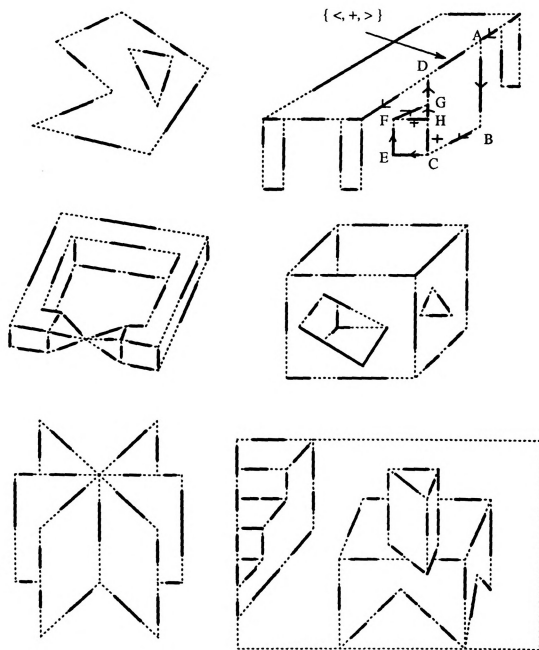
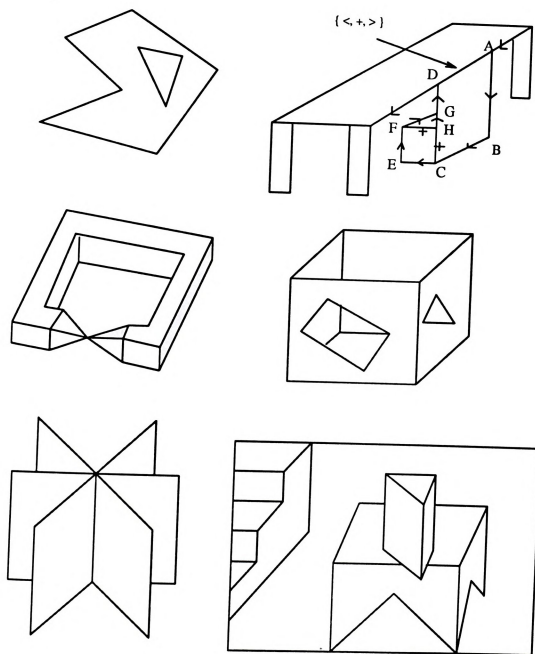


Figure 4.18. Reconstruction of the Big Block. (a) Detected wings (solid lines). (b) Individual reconstructed faces. (c) Reconstructed LLDG.



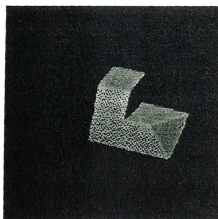
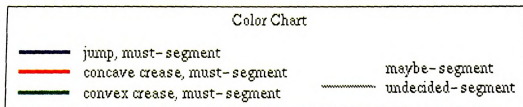
(a) wing samples

Figure 4.19. cont. (see page 135 for caption)

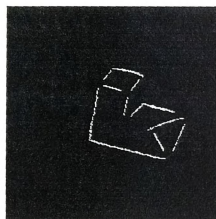


(b) reconstructed LDG

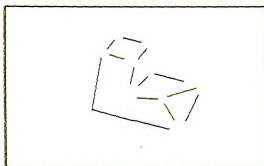
Figure 4.19. More examples of syntactic scenes that have been reconstructed. (a) Wings samples. (b) Reconstructed LDG.



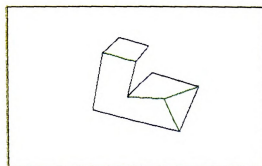
(a) original intensity image



(b) detected wing samples



(c) augmented wing samples



(d) reconstructed LLDG

Figure 4.20. Reconstruction of the LLDG of a view of the gb3 block.

is decomposed into analysis of those individual components. Of those four modules, the two algorithms only differ in how the undecided-segments are resolved. In the following analysis, we assume that the number of wing samples is n .

Wing Clustering

The wing clustering step is simplified with the ideal assumption about the measurement and the computational accuracy. Co-planar wings all have the same wing-surface labels on one of the two wing-surfaces; therefore, only one pass through the wing set is necessary to form the co-planar wing clusters. Suppose wing $w_1 \dots w_{i-1}$ have already been placed into m proper clusters, where $m \leq 2(i-1)$ (recall that each wing must belong to exactly two planar clusters). To place wing w_i into the two proper planar clusters, at most m comparisons with the already existing planar clusters are needed. Thus the complexity of the clustering step is

$$O\left(\sum_{i=1}^n 2(i-1)\right) = O(n^2).$$

Wing Extensions and Intersections

Assume that there are m different wing clusters, C_1, C_2, \dots, C_m with n_1, n_2, \dots, n_m wings, respectively. Note that $n_1 + n_2 + \dots + n_m = 2n$ and the minimum number of clusters is 2. For each wing cluster C_i , the wings are extended and intersected with other wing extensions in the cluster which is accomplished in $O\left(\binom{n_i}{2}\right) = O(n_i^2)$ time. Summing over all wing clusters, the total time is

$$\sum_{i=1}^m O\left(\binom{n_i}{2}\right) = \sum_{i=1}^m O(n_i^2).$$

Since $\binom{n_i}{2} > 0, \forall i$, it is easy to see that the above sum is bounded above by

$$O((n_1 + n_2 + \dots + n_m)^2) = O((2n)^2) = O(n^2)$$

This worst case scenario only occurs when all the wing samples are co-planar giving rise to two planar groups: the plane of the object faces and the background.

Resolving Undecided-Segments

Initially, the number of undecided-segments in each wing cluster is at most n_i^2 . Using POLY-1, each of those segments is visited only once to either change it into a must-segment or an unwanted-segment. Note that once a segment is "decided" it can never be changed back to the undecided state. Therefore the time complexity over all wing clusters is

$$\sum_{i=1}^m O(n_i^2),$$

which again is bounded above by

$$O((n_1 + n_2 + \dots + n_m)^2) = O((2n)^2) = O(n^2)$$

With POLY-2, the algorithm iterates over the set of remaining undecided-segments until either no more undecided-segments remain or no more updating is possible. Thus, during each pass through the remaining undecided-segments, at least one undecided-segment is permanently updated. The worst case scenario occurs when only one undecided-segment changes state (to either must or unwanted) during each iteration (in practice, several undecided-segments would have been updated during each pass). Thus for wing cluster C_i , which has at most n_i^2 undecided-segments, the updating will stop after at most n_i^2 iterations and the number of remaining undecided-segments is decremented by at least one after each iteration. Therefore the worst case

time complexity is characterized by

$$O(\sum_{j=1}^{n_i^2} j) = O(\frac{(1 + n_i^2)(n_i^2)}{2}) = O(n_i^4).$$

Over all wing clusters, the total time complexity is

$$\sum_{i=1}^m O(n_i^4),$$

which is bounded above by

$$O((n_1 + n_2 + \dots + n_m)^4) = O((2n)^4) = O(n^4).$$

Merging of Planar Faces

In merging planar faces, must-segments from two non-co-planar faces are intersected in 3D to discover the final line segments in the LDG and to compute the 3D vertices. Recall that the number of line segments in each planar group of faces is bounded by the square of the number of wings in the planar group. Theoretically, for group C_i , at most n_i^2 line segments can appear in the reconstructed face group C_i . Over all planar groups, the upper bound on total number of line segments would be $(n_1^2 + n_2^2 + \dots + n_m^2)/2$ (each segment must appear in two planar groups). However, the ideal wing sample assumptions A_3 and A_4 restrict the number of line segments in the final LDG to be at most the number of wings samples, n . Thus, there can only be at most $2n$ line segments among all the reconstructed faces. Pairwise intersecting those segments yields a worst case time complexity of

$$\sum_{i=1}^{2n} O\left(\binom{2n}{2}\right) = O(n^2).$$

Overall Worst Case Time Complexities

For the POLY-1 algorithm, the worst case time complexity is

$$O(n^2) + O(n^2) + O(n^2) + O(n^2) = O(n^2).$$

For the POLY-2 algorithm, the worst case time complexity is

$$O(n^2) + O(n^2) + O(n^4) + O(n^2) = O(n^4).$$

4.7 Summary

We have given two algorithms for reconstructing line drawings from “wing samples” for a large class of polygonal scenes and views. The mathematical characterization of legal scenes and views was given in Chapter 2, which we believe to be more general and more practical than those in previous works. A junction catalog was not assumed, nor were accidental views ruled out by topological events. Moreover, fallible heuristics based on principles of human perceptual organization have not been used. Both theoretical and practical considerations indicate that wing features are quite appropriate for low-level representation of polygonal scenes.

In POLY-1, we resampled the range values during reconstruction. In step (2.4), a limited search of the range image was performed in order to get information that was not directly coded in the bottom-up wing representation. This step allows the processing to continue to completion using only local processing. Via POLY-2, we showed that the line drawing can actually be constructed without such further access to the underlying range image. The second algorithm searches for valid interpretations of each surface using general geometrical constraints. Since the rules employed were shown to be necessary, no valid interpretations will ever be thrown away. The

surface merging procedure has successfully merged and resolved any ambiguous surface interpretations in our test cases.

Both algorithms presented are suitable for parallel implementation. Once the wing groups are decided, reconstruction of each surface can proceed independent of one another. Parallel implementation of the second algorithm can also cure the problem of ambiguous surface interpretations by communicating with adjacent plane groups for information about the undecided segments.

All of the results presented in this chapter hinge on strong assumptions about the input wing set and accuracy of computation. In the next chapter, we will present a heuristic algorithm that can better deal with imperfect set of wing samples.

CHAPTER 5

Partial Reconstruction of LLDG - Origami/Polyhedral World

5.1 Introduction

The algorithms depicted in the previous chapter rely strongly on the assumptions that the wing samples are complete and that the computations are infinitely accurate. Those assumptions are unrealistic in practice and were asserted only to facilitate a better understanding of the reconstruction algorithms. The algorithm to be given in this chapter is heuristic based. Heuristic rules are used to replace the deterministic rules and to account for those unrealistic assumptions. Only the two object domain assumptions from Chapter 4 survive the reality check in this chapter:

- A_1) Objects are δ -polygons or polyhedra whose faces are δ -polygons.
- A_2) Scenes can be composed of multiple objects which may occlude one another.

We do not make any assumptions about the completeness of wing samples; an imperfect set of wing samples with missing and/or spurious wings is allowed as input. Furthermore, the non-accidental view assumption is also omitted. From the definition

of accidental view in Chapter 2, wings may not be sensed on some of the visible edges in an accidental view, which results in missing wings.

In the next section, we briefly review past and present work on line drawing reconstruction. This is followed by the presentation of our approach. A good heuristic algorithm should result in a complete LLDG if the input wing samples are complete. We show by example that this is indeed the case. Examples of partially reconstructed LLDGs from imperfect sets of wings are also given.

5.2 Survey of Related Background

Construction of *perfect* line drawings from a raw input image or imperfect line drawing has only been sparingly reported in the literature. Although an abundance of work has been done in image segmentation, existing edge detection techniques are still far from being perfect. Furthermore, line labeling algorithms surveyed in the previous chapter all require that all junctions in the image be identified.

Perhaps the earliest documented work on obtaining line drawings of polyhedral scenes is that of Shirai [111]. The input is an intensity image from which the line drawing is simultaneously generated and interpreted. Shirai assumes that the scenes consist of only polyhedral objects and are always set up with high contrast between white objects and their black background. Thus the contour lines in the image are easy to find. The strategy for finding the lines is to first extract the contour lines in the image and then to heuristically search along the contour for other object boundary lines and crease edge lines. The set of heuristics used not only finds lines, but also facilitates the finding of the relationships between objects. When a complete set of boundary lines is found, all the objects can then be identified. However, this top-down approach for line finding may fail to generate the appropriate line drawing

if some contour/boundary lines were not detected or if spurious lines were found due to noise in the image, especially during the early stages of the process.

In [49], Gu and Huang described a heuristic-search algorithm for extracting a connected line drawing from a perspective view of a polyhedral scene. The main feature of the algorithm is that instead of detecting lines (edges) in the image before forming the junctions (corners), the junctions are located first and the search for lines concentrated on local areas centered at each junction. An image of not necessarily connected edge points is obtained by thinning the thresholded Sobel gradient image. A corner finder routine [36] is then applied to this sparse edge map to generate a list of plausible corners. Final edges in the image are found by heuristically searching for evidence that an edge exists between two corners for every possible pair of corners. A drawback of this approach is that corner finding is expensive and difficult. Furthermore, if a visible corner is not detected, all line segments incident with that corner will be lost.

Most recently, Trytten has proposed to create a labeled line drawing graph from a single intensity image by integrating multiple perceptual modules [119]. The approach is to take existing low level perceptual modules and integrate them through the use of a blackboard system as a common database, where each module can cooperate and compete with others to search for a good interpretation of the input image. The modules proposed for integration include:

1. regularization based surface segmentation module [15],
2. boundary detection via perceptual grouping [120],
3. line labeling analysis [83], and
4. symmetry detection module [47].

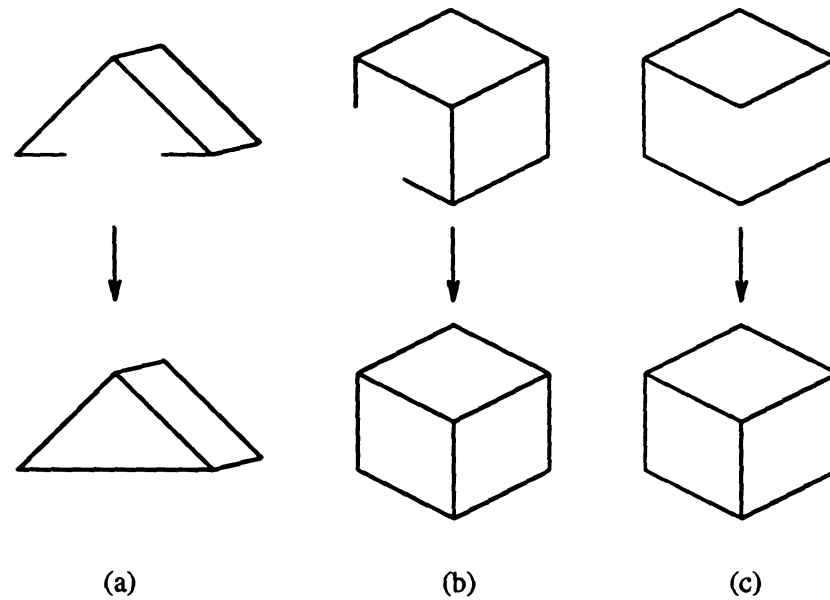


Figure 5.1. Falk's heuristic rule for completing imperfect line drawings [34].

This approach might be robust since features that are hard to detect and consequently missed by one module may be diagnosed as being imperfect and possibly fixed by another module. However, the complexity of integrating various modules may be overwhelming and must be overcome.

Instead of deriving an entire line drawing from scratch, Falk [34] assumes that an incomplete line drawing is obtained, fills in the missing line segments based on heuristic rules, and finally uses object models to aid the interpretation of the line drawing. Three ad hoc rules are used to complete the line drawings:

1. Two dangling collinear lines are replaced by a single line. (Figure 5.1 (a))
2. If two dangling lines on an incomplete face can be extended to form a corner, then the lines are extended to complete the face. (Figure 5.1 (b))
3. If there exists a pair of L-type junctions with parallel sides as in Figure 5.1 (c), then add a line between these two junctions to split the face into two.

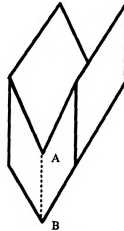


Figure 5.2. Spurious line could be added with Falk's [34] rules.

Obviously those rules may result in spurious line segments in many real life situations. For example, Rule 3 would unnecessarily introduce a new segment between junctions A and B in Figure 5.2. Based on the amended line drawing, Falk's algorithm hypothesized about objects in the scene and their orientations, then predicted the line drawing of the hypothesized scene, and finally tried to verify it.

5.3 Heuristic LLDG Reconstruction Algorithm

The flow of the heuristic based algorithm is very similar to the two deterministic algorithms presented in the previous chapter. A quick overview of the algorithm is as follows. First, some preprocessing is needed to screen out spurious wings and to cut down the number of wings to use for reconstruction. The remaining wings are clustered according to their wing-surface labels. This step is no longer trivial because we have removed the measurement and computation accuracy assumption. As before, each cluster of wings is processed independently. Wings within each cluster are extended and intersected as before. Each line segment is labeled using one of the six heuristic rules. The notion of *maybe-segment* is introduced to facilitate the recon-

struction process. After (partial) reconstruction of all the planar faces, they must be merged to form the final LDG, which involves more heuristic decision making. The complete algorithm as well as the major steps that are different from the algorithms of Chapter 4 are given and discussed in detail in the following subsections.

5.3.1 Preprocessing of Wing Samples

Using the detection algorithm depicted in Chapter 3, the sampled wings are often short and overlapping. Multiple wings extracted from the same physical edge often have the same wing label; however, due to measurement and computation inaccuracies, those wings almost never have the same set of parameter values. Since a line segment in the LDG can be recovered if *at least* one wing is present, it is advantageous to combine those overlapping compatible short wings into one long wing before performing LDG reconstruction. Reconstructing the LDG using the new set of longer wings reduces the complexity of the algorithm.

Basically, two wings are combined into one if they overlap and the wing attributes are compatible. Two wings are compatible if their wing-contour labels and the two wing-surface labels are equivalent; furthermore, the wing-contours in 2D must be near-collinear within some threshold. Note that an edge may still be represented by more than one non-overlapping wing after the merge.

Another task undertaken by the preprocessing module is to screen out spurious wings. Spurious wings are often the result of surface marking. While detection of surface marks may be of some importance to other vision tasks, this research strives to reconstruct the shape of the objects in the scene but not the features on the surface of those objects. Thus those wings shall be dismissed before the actual reconstruction takes place. A common trait among those spurious wings is that the surface labels (equation) on both sides of the wing are nearly coplanar. Experiments have shown that usually only a few spurious wings were detected by the wing detector described

in Chapter 3. In those rare instances, spurious wings often can be screened out by the above procedure.

5.3.2 Clustering of Wings

As in the POLY-1 and POLY-2 algorithms, the first step in the reconstruction phase is to cluster wings into groups according to their wing-surface labels. Each wing belongs to exactly two planar groups (counting background or don't care groups as one planar group). Previously, infinite measurement accuracies were assumed and the planes were denoted symbolically which assured that coplanar wings have common wing-surface labels. Given this assumption, the clustering process was a trivial task. In reality, the wing-surface labels are the parameters of the planar surfaces as detected by the wing detector which may not be exactly the same even for coplanar wings.

Our clustering strategy is motivated by the pose clustering technique used by Stockman [114]. First, the plane equations from all wings are collected into a pool, say \mathbf{P} . Plane equations in \mathbf{P} are then pairwise clustered. A pair of planes are equivalent and therefore clustered if they are coplanar in the sense that the angle between their surface normals and the difference in distances to the origin are within some predetermined thresholds. Clusters of planar groups are formed such that members of each plane cluster $\mathbf{P}_i = \{p_{i1}, p_{i2}, \dots\}$ are either symmetrically equivalent ($p_{ij} \equiv p_{ik}$) or transitively equivalent ($p_{ij} \equiv p_{ik} \equiv \dots \equiv p_{im}$). Finally, wings with wing-surface labels in the same cluster are collected as a wing cluster. A symbolic label is assigned to each plane cluster and the wing-surface labels are replaced with those symbolic symbols.

Having formed planar wing clusters, the next step is to recover the planar faces and their boundaries within each cluster. Wings are extended and the intersections of the extended wings are found as before. The junctions are taken to be the union of

those intersections and the actual wing segment endpoints. The initial must-segments are those segments overlapped by a wing segment. The rest were undecided-segments.

5.3.3 Notion of Maybe-Segments

Before introducing the heuristic rules, we must add a new segment type. Previously, an undecided-segment became a must-segment if some criteria were satisfied. Here, an undecided-segment must first become a *maybe-segment* before becoming a must-segment. The strategy for finding all the must-segments is to extend the existing must-segments whenever possible. The maybe-segments are the segments that are subjected to the must-segment tests. So the sole purpose of this extra stage is to screen for the most likely candidates for the tests. A maybe-segment must be an extension of an existing must-segment; furthermore, its adjacent region labels must be consistent with those of its neighboring must-segments if they exist.

Let P_δ denotes the plane equation of the planar group that is being reconstructed. An undecided-segment seg_{ij} becomes a *maybe-segment* if

1. seg_{jk} is a must-segment and is collinear with seg_{ij} ; and
2. $SegCCW_{v_j}(seg_{ij}) (SegCW_{v_j}(seg_{ij}))$ is a must-segment and the region labels for the region shared by seg_{ij} and $SegCCW_{v_j}(seg_{ij}) (SegCW_{v_j}(seg_{ij}))$ contains P_δ .

Circumstances in which an undecided-segment is and isn't turned into a maybe-segment are depicted in Figure 5.3.

5.3.4 Heuristic Rules

Reconstruction of faces in each planar group of wings proceeds independently. For convenience, we denote the planar group under reconstruction as P_δ (i.e., P_δ is the common wing-surface label of all wings in the group).

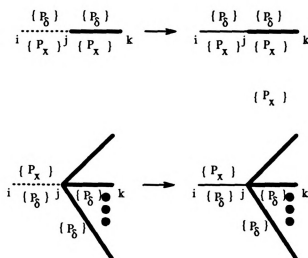
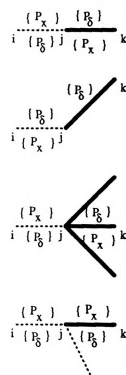
Segment \overline{ij} a maybe-segmentSegment \overline{ij} not a maybe-segment

Figure 5.3. Some examples of *Maybe*-segments and non-*Maybe*-segments. Notation: dashed segments are undecided-segments; thin solid segments are maybe-segments; thick solid segments are must-segments.

Since our experiments have shown that occurrence of spurious wings are rare and when they do occur, the preprocessing step would remove them, we will proceed as if no spurious wings survive to this level. Thus no line segments are results of extension of spurious wings. However, there may still be missing line segments due to missing wings. From this observation, Rule 4.1 of the POLY-2 algorithm can also be applied here. Specifically, if a junction is incident with an even number of must-segments and if only one undecided-segment is incident with that junction, then that undecided-segment shall be removed as an unwanted-segment.

Rule 5.1 *An undecided-segment or maybe-segment $seg_{ij} \equiv (v_i v_j)$ is discarded as unwanted-segment if $deg_{must}(v_i) = \text{even}$ and $deg_{und}(v_i) = 1$; or if $deg_{must}(v_j) = \text{even}$ and $deg_{und}(v_j) = 1$.*

Note that Rule 4.2 used by algorithm POLY-2 to make an undecided-segment a must-segment cannot be used because line segments could be missing due to missing wings.

The first condition in which a maybe-segment can become a must-segment is based on the Gestalt Law of Good Continuation [72, 124] and is graphically illustrated in Figure 5.4.

Rule 5.2 *Given line segments seg_{ij} , seg_{jk} , and seg_{kl} are collinear with compatible attributes where seg_{ij} and seg_{kl} are must-segments and seg_{jk} is a maybe-segment. Then seg_{jk} becomes a must-segment if junctions j and k are coincident with no other segments.*

Basically, Rule 5.2 fills the void between two collinear must-segments to form one continuous line that would otherwise be two dangling must-segments.

While Rule 5.2 fills in the gap on a broken but likely continuous line, the next two rules reconstruct the junctions of each face with high degree of certainty. The situations in which those rules can be applied are graphically illustrated in Figure 5.4.

Heuristic Rule 2

$$\frac{\overline{i \begin{matrix} \{P_0\} \\ \{P_x\} \end{matrix} j \begin{matrix} \{P_0\} \\ \{P_x\} \end{matrix} k \begin{matrix} \{P_0\} \\ \{P_x\} \end{matrix} l}}{\longrightarrow} \overline{i \begin{matrix} \{P_0\} \\ \{P_x\} \end{matrix} j \begin{matrix} \{P_0\} \\ \{P_x\} \end{matrix} k \begin{matrix} \{P_0\} \\ \{P_x\} \end{matrix} l}$$

Heuristic Rule 3

$$\begin{array}{ccc} \begin{array}{c} \text{Dashed lines from } j \\ \text{Dashed lines from } i \\ \text{Dashed lines from } k \end{array} & \longrightarrow & \begin{array}{c} \text{Dashed lines from } j \\ \text{Dashed lines from } i \\ \text{Dashed lines from } k \end{array} \\ \begin{array}{c} \{P_x\} \quad \{P_x\} \\ \{P_0\} \quad \{P_0\} \end{array} & & \begin{array}{c} \{P_x\} \quad \{P_x\} \\ \{P_0\} \quad \{P_0\} \end{array} \end{array}$$

Heuristic Rule 4

$$\begin{array}{ccc} \begin{array}{c} \text{Dashed lines from } j \\ \text{Dashed lines from } i \\ \text{Dashed lines from } k \\ \text{Dashed lines from } h \end{array} & \longrightarrow & \begin{array}{c} \text{Dashed lines from } j \\ \text{Dashed lines from } i \\ \text{Dashed lines from } k \\ \text{Dashed lines from } h \end{array} \\ \begin{array}{c} \{P_x\} \quad \{P_x\} \\ \{P_0\} \quad \{P_0\} \end{array} & & \begin{array}{c} \{P_x\} \quad \{P_x\} \\ \{P_0\} \quad \{P_0\} \end{array} \end{array}$$

Heuristic Rule 5

$$\begin{array}{ccc} \begin{array}{c} \text{Dashed lines from } j \\ \text{Dashed lines from } i \\ \text{Dashed lines from } k \\ \text{Dashed lines from } h \end{array} & \longrightarrow & \begin{array}{c} \text{Dashed lines from } j \\ \text{Dashed lines from } i \\ \text{Dashed lines from } k \\ \text{Dashed lines from } h \end{array} \\ \begin{array}{c} \{P_x\} \quad \{P_x\} \\ \{P_0\} \quad \{P_0\} \end{array} & & \begin{array}{c} \{P_x\} \quad \{P_x\} \\ \{P_0\} \quad \{P_0\} \end{array} \end{array}$$

Heuristic Rule 6

$$\begin{array}{ccc} \begin{array}{c} \text{Dashed lines from } j \\ \text{Dashed lines from } i \\ \text{Dashed lines from } k \\ \text{Dashed lines from } h \end{array} & \longrightarrow & \begin{array}{c} \text{Dashed lines from } j \\ \text{Dashed lines from } i \\ \text{Dashed lines from } k \\ \text{Dashed lines from } h \end{array} \\ \begin{array}{c} \{P_x\} \quad \{P_x\} \\ \{P_0\} \quad \{P_0\} \end{array} & & \begin{array}{c} \{P_x\} \quad \{P_x\} \\ \{P_0\} \quad \{P_0\} \end{array} \end{array}$$

Figure 5.4. Graphical illustration of Heuristic Rules 2-6.

Rule 5.3 *Given noncollinear line segments seg_{ij} , seg_{jk} , and seg_{kl} where seg_{ij} and seg_{kl} are must-segments while seg_{jk} is a maybe-segment that is collinear with seg_{kl} . In addition, seg_{ij} and seg_{jk} are neighboring segments around junction j . If*

$$RL-CCW_j(seg_{ij}) = \{P_\delta\} \text{ and } P_\delta \in RL-CW_j(seg_{jk})$$

or

$$RL-CW_j(seg_{ij}) = \{P_\delta\} \text{ and } P_\delta \in RL-CCW_j(seg_{jk}),$$

then seg_{jk} becomes a must-segment and the extensions (connected undecided-segments from junction j) of these two segments into the non-shared region(s) are removed as unwanted-segments.

Rule 5.4 *Given two pairs of collinear compatible segments $\{seg_{hi}, seg_{ij}\}$ and $\{seg_{jk}, seg_{kl}\}$ where seg_{hi} and seg_{kl} are must-segments while seg_{ij} and seg_{jk} are maybe-segments. In addition, seg_{ij} and seg_{jk} are neighboring segments around junction j . If*

$$P_\delta \in RL-CCW_j(seg_{ij}) \text{ and } P_\delta \in RL-CW_j(seg_{jk})$$

or

$$P_\delta \in RL-CW_j(seg_{ij}) \text{ and } P_\delta \in RL-CCW_j(seg_{jk}),$$

then both seg_{ij} and seg_{jk} become must-segments and the extensions (connected undecided-segments from junction j) of these two segments into the non-shared region(s) are removed as unwanted-segments.

In the case of Rule 5.3, the fact that a must-segment is coincident with the maybe-segment at a junction and that they both label the region they share as P_δ gives strong suggestion that the maybe-segment indeed is a must-segment. Rule 5.4 differs only in that two maybe-segments meet at the junction instead of one must- and one maybe-segment. However, Rule 5.4 is not weaker than Rule 5.3 because the maybe-segments are all immediate extension of must-segments and the evidence from wing-surface labels supports the hypothesis those maybe-segments should both be must-segments.

While the first 4 rules can be applied with high degree of confidence, they are not sufficient in reconstructing all of the image scenes that were tested. The next two rules are the weaker version of Rules 5.3 and 5.4. They are weaker because the maybe-segments are still going to become must-segments even when other undecided-segments are present in the region that is labeled as P_δ by both segments. Example of their usage is given in Figure 5.4. Even though applying these two rules *may* introduce untrue must-segments, the risk is minimized by invoking them only when no more segments can be decided by any of other rules. Moreover, a must-segment on a particular planar group of faces does not automatically carry over to the final LDG. The same must-segment must appear in one other planar group of faces or it maybe downgrade to a maybe-segment in the final LDG. Merging of planar faces is discussed in the next subsection.

Rule 5.5 *Given noncollinear line segments seg_{ij} , seg_{jk} , and seg_{ki} where seg_{ij} and seg_{ki} are must-segments while seg_{jk} is a maybe-segment that is collinear with seg_{ki} . Looking at a small neighborhood N_r of radius $r < \epsilon$, Line segments seg_{ij} and seg_{jk} locally divide N_r into two regions. If*

$$RL-CCW_j(seg_{ij}) = \{P_\delta\} \text{ and } P_\delta \in RL-CW_j(seg_{jk})$$

or

$$RL-CW_j(seg_{ij}) = \{P_\delta\} \text{ and } P_\delta \in RL-CCW_j(seg_{jk})$$

and there are only undecided-segments in the region in which the above conditions were met, then seg_{jk} becomes a must-segment and the extensions (connected undecided-segments from junction j) of these two segments into the other region are removed as unwanted-segments.

Rule 5.6 Given two pairs of collinear compatible segments $\{seg_{hi}, seg_{ij}\}$ and $\{seg_{jk}, seg_{kl}\}$ where seg_{hi} and seg_{kl} are must-segments while seg_{ij} and seg_{jk} are maybe-segments. Looking at a small neighborhood N_r of radius $r < \epsilon$, Line segments seg_{ij} and seg_{jk} locally divide N_r into two regions. If

$$P_\delta \in RL\text{-}CCW_j(seg_{ij}) \text{ and } P_\delta \in RL\text{-}CW_j(seg_{jk})$$

or

$$P_\delta \in RL\text{-}CW_j(seg_{ij}) \text{ and } P_\delta \in RL\text{-}CCW_j(seg_{jk}),$$

then both seg_{ij} and seg_{jk} become must-segments and the extensions (connected undecided-segments from junction j) of these two segments into the other region are removed as unwanted-segments.

The rules are applied from 1 to 6 in that order since the first four rules have more local evidences in turning undecided/maybe-segments into unwanted and/or must-segments. Upon application of each rule, a check is performed to remove any false junction (junction that appeared on collinear must-segments). If a false junction is detected and removed, the extension of any must/maybe-segment incident with that false junction is also updated to reflect this change. Afterward, the cycle repeats starting from Rule 5.1. The process stops when there is no more undecided- or maybe-segment to decide or when none of the rules can be applied to the remaining undecided/maybe-segments.

5.3.5 Merging of Planar Faces

Once all the faces have been (partially) reconstructed, they need to be merged together to arrive at the final LLDG of the scene. The merging process is not as trivial as before because some of the faces now may still have missing lines after reconstruction. Although a line segment theoretically should appear on exactly two faces, missing wings can cause a line segment not to appear in one or both faces. Further-

more, even if it does appear in both faces, it may be labeled differently (*e.g.*, labeled as *must* in one and as *maybe* in the other). More heuristic steps must be taken to ensure that each segment in the final LLDG has a unique interpretation in the sense that it is either a *must*-segment or *maybe*-segment. Note that a line segment is said to appear on a face only if it is labeled either as a *must*-segment or as a *maybe*-segment.

The first step in merging all the faces is to pull together all the *must*- and *maybe*-segments and determine all the intersections of all those segments. Three very common problems encountered as a result of this operation are depicted in Figure 5.5 (a). All of these problems can be attributed to computation and measurement inaccuracies. First, many short residual segments may result around the intersection of two segments. On the other hand, two lines that were supposed to intersect may fail to do so. Thirdly, when three or more supposedly co-junction line segments meet, instead of meeting at one unique junction, three or more different close-by junctions will occur. To alleviate these problems, after determining all the junctions in the LDG, nearby junctions are clustered into one unique junction. All the line segments that terminate in the same cluster of junctions have their endpoints replaced by that unique junction. In so doing, the overshoot of line segments are removed; the gaps between two supposedly intersecting line segments are filled; and the multi-junctions are merged into one (see Figure 5.5 (b)).

Once all the junctions have been determined, we must resolve any ambiguity in labeling each line segment (*i.e.*, *must*- or *maybe*-segment). If the line fragment appears on two faces and it is labeled as a *must*-segment on at least one of the two faces, then the segment is labeled as a *must*-segment. Otherwise, the line fragment is labeled as a *maybe*-segment. This is the case where the line fragment is labeled as *maybe*-segment on both faces.

When the line fragment only appears on one face and it is labeled as a *must*-segment, then the decision process is more complicated. Recall that during recon-

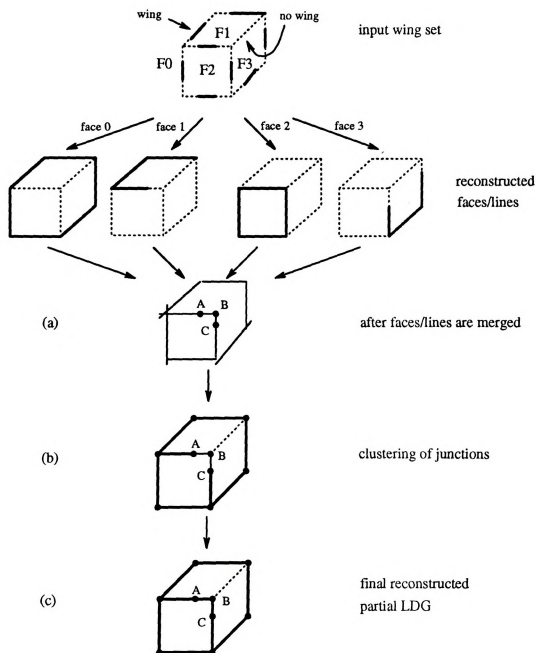


Figure 5.5. Problems encountered when merging faces into final LDG their solutions

struction of a face, the original set of must-segments are the actual wing segments which are just *fragments* of the real line segment in the LDG. Let's denote the endpoints of those wing segments as wing endpoints. If the reconstructed face contains a dangling wing segment, it would indicate that some line segment(s) of the face is (are) missing because that wing is not extended to meet with other line segments. Therefore a wing endpoint in the reconstructed face indicates that the wing segment should be extended further at that end. Thus, a line fragment in the merged LDG which appears on only one face would be labeled as a must-segment if (1) it is labeled as a must-segment on that single face and (2) one of its endpoints is a wing endpoint. Otherwise, that line fragment is labeled as a maybe-segment. However, if the segment is labeled as a maybe-segment on the only face in which it appears, then it is discarded as an unwanted-segment. Examples of different cases mentioned above are depicted in Figure 5.5 (c).

Upon resolution of the line labels, the LLDG is finally reconstructed. Since some line segments may still be missing due to missing wings, the reconstructed LLDG shall be called a partial LLDG.

5.3.6 Heuristic Reconstruction Algorithm: POLY-3

The entire reconstruction procedure is summarized below. The input is a set of sampled wings which may contain spurious and/or missing wings. The output is a (partial) LLDG whose completeness is directly related to the completeness of the sampled wings. Examples of scenes that can be handled are given in the next section. A list of the major parameters used in this algorithm is given in Appendix C.

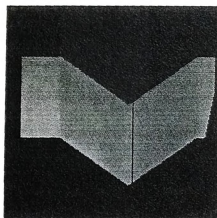
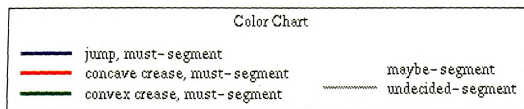
- (0) Preprocess wing samples to merge compatible wings and to cull out extraneous wings.
- (1) Cluster merged wing samples by plane equation

- (2) For each plane group
 - (2.1) generate a line for each wing
 - (2.2) compute the intersections among all lines
 - (2.3) preserve line segments which overlap a wing
 - (2.4) remove or preserve line segments by using Rules 1, 2, 3, 4, 5 and 6.
 - (2.4.1) if $V_{E_1} \neq \text{null}$, apply Rule 1 then goto (2.4.1)
 - (2.4.2) apply Rule 2. If invoked, goto (2.4.1)
 - (2.4.3) apply Rule 3. If invoked, goto (2.4.1)
 - (2.4.4) apply Rule 4. If invoked, goto (2.4.1)
 - (2.4.5) apply Rule 5. If invoked, goto (2.4.1)
 - (2.4.6) apply Rule 6. If invoked, goto (2.4.1)
 - (2.4.7) no more changes are possible, goto (2.5)
- (3) Merge LDGs of individual polygonal faces
- (4) Compute 3D vertices using intersecting line/plane equations
- (5) Retain all line segment labels from wings
- (6) Output LDG with plane equations, 3D vertices and edges

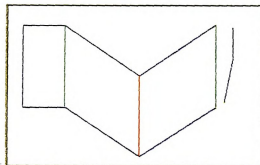
5.4 Examples

A complete example depicting the major phases of the reconstruction process is given in Figure 5.6. The set of wing samples as detected by the wing sensor is imperfect. It consists of some spurious wings and missing wings. Many close-by parallel wings are also present in the wing set. After the initial wing merging step, all the spurious wings are removed and the compatible wings are merged; only 13 wings remain for the reconstruction. The initial and final states of each wing cluster upon reconstruction are shown in Figure 5.6 (e)-(l). Note that some false must-segments are inserted in (e) and (f). However, each is downgraded to a maybe-segment due to non-sufficient evidence from the other faces. The final reconstructed LLDG is depicted in (b).

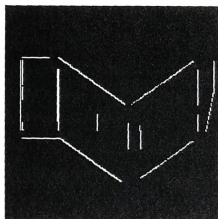
More LLDG reconstruction examples are shown in Figure 5.7. The images on the left column show the sampled wings overlaid on the original intensity image while the



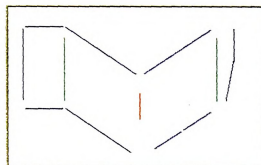
(a) original image



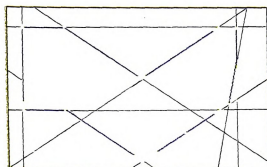
(b) reconstructed LLD



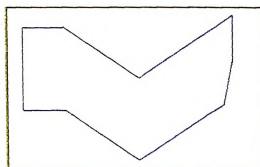
(c) detected wing samples



(d) merged wings



(e) wing cluster 0

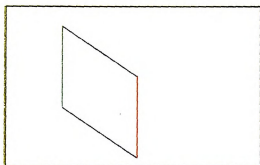


(f) reconstructed cluster 0

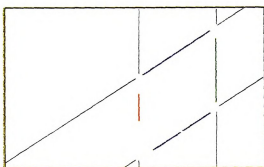
Figure 5.6. cont. (see page 161 for caption)



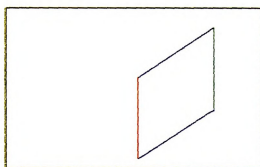
(g) wing cluster 1



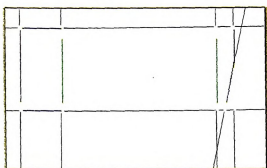
(h) reconstructed cluster 1



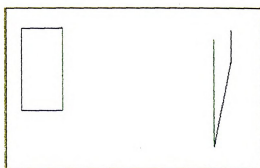
(i) wing cluster 2



(j) reconstructed cluster 2



(k) wing cluster 3



(l) reconstructed cluster 3

Figure 5.6. Reconstruction of the JIG via algorithm POLY-2

right column shows the reconstructed LLDG. Since we have assumed that only planar surfaces can appear in the scene, the wing detector was tailored to detect only planar wings. The “roof1” example is the only one in which there was no missing wing and the complete LLDG was reconstructed. The others all have defective wing samples. We must also admit that some of the wings in the “jig1+box1” did not cluster properly and were manually placed in the “correct” cluster before proceeding. The missing wings in the “gb3” and “block1” examples are direct effects of the quantized wing detecting window. In both cases, whenever the window overlapped the edge contour with no wing, it also overlapped some other junction, resulting in a busy window. As previously discussed in Chapter 3, a scale-space approach to wing detection may cure this problem. Furthermore, in the “gb3” example, although a face is completely missing (having no range values due to occlusion of the laser light), an LLDG was nevertheless reconstructed from the available wing samples. Note that line labeling of the reconstructed LDG using the Huffman-Clowes junction catalogue would not be possible due to the missing face. The missing wings in the “planar2” and “blocks1” examples were due to slight mis-registration of range and intensity data: the intensity data and range data gave conflicting information about the presence of a wing, which resulted in having no wing detected.

Table 5.1 gives the performance summary of the POLY-3 algorithm in reconstructing the LLDGs of the real images in Figures 5.6 and 5.7. Observe that in the “block1” image, an arc in the ideal LLDG with no wing sample was correctly inferred. However, one must-segment was also incorrectly inferred, showing the fallible heuristic nature of the algorithm. Upon closer examination, we found that there was some evidence for the existence of that extra must-segment and no evidence to discourage its formation. Furthermore, its existence was tied to the fact that an arc that was supposed to adjoin was missing in the reconstructed LDG due to missing wing samples on that arc. For all other images, all arcs with at least one wing sample were correctly reconstructed

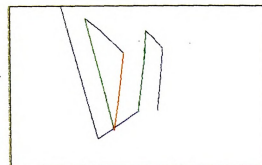
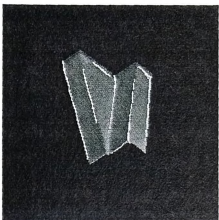
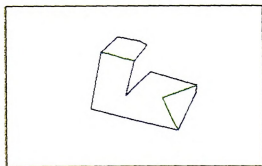
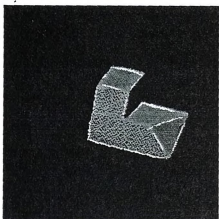
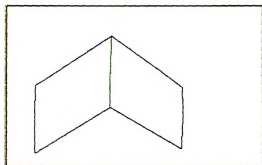
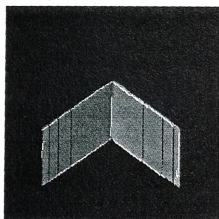
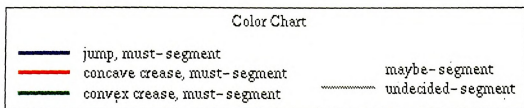


Figure 5.7. cont. (see page 164 for caption)

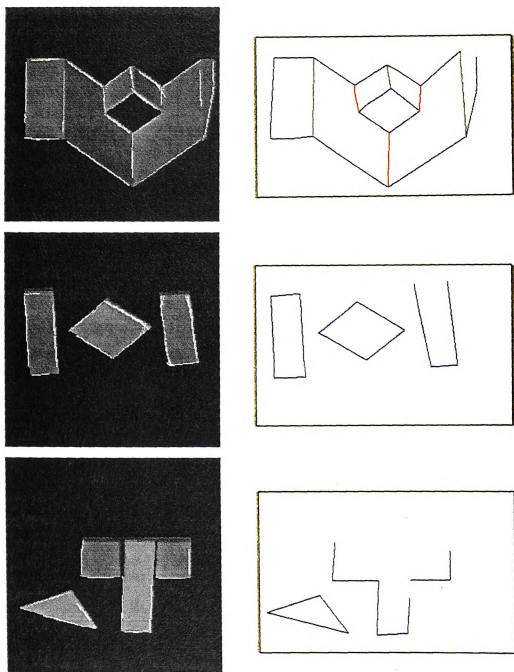


Figure 5.7. Examples of reconstructed LLDG via algorithm POLY-2 (Names of the scenes: *roof1*, *gb3*, *block1*, *jig1+box1*, *planar2*, *blocks1*).

and no false arcs were introduced in the reconstructed LLDGs. Associated with the missing arcs are the incomplete regions in the reconstructed LLDGs. However, most of the incomplete regions have only one missing arc. Despite the incompleteness of the LLDGs, the features embedded in the partially reconstructed LLDGs seem to be rich enough for any reasonable object recognition module to infer the objects from the model database.

Finally, we make a note that the algorithm has been applied to the synthetically generated test cases of Chapter 4, where the wing samples were complete. The algorithm is able to reconstruct the complete LLDG as would the previous algorithms in those test cases.

5.5 Worst Case Time Complexity of POLY-3

As with the POLY-2 algorithm, POLY-3 also consists of the same four major sequential modules: clustering of wing samples, extending and intersecting wings, resolving the undecided-segments and merging of the planar faces. Again, assume that the number of wing samples after the preprocessing step is n (Note the preprocessing module requires $O(w^2)$ time to complete where w is the original number of wing samples).

Wing Clustering

Recall that the wing clustering step was performed by first pairwise clustering the wing samples followed by distribution of wings into clusters such that wings within each cluster are either symmetrically equivalent or transitively equivalent. Since there are n wing samples, the pairwise clustering step can be completed in

$$O\left(\binom{n}{2}\right) = O(n^2)$$

Table 5.1. Summary of the LLDG of the 7 real images.

Scene	input			output					
	ideal # of arcs in the LLDG	# of arcs having no wing samples	ideal # of regions in the LLDG	# of correct must- segments	# of wrong must- segments	# of maybe- segments	# of missing arcs	# of complete regions	# of regions missing one arc
jig1	14	2	4	12	0	5	2	3	0
roof1	7	0	2	7	0	0	0	2	0
green block	13	1	4	12	0	0	1	2	2
block1	12	4	4	9	1	1	3	1	3
jig1+box1	22	1	6	21	0	2	1	5	1
planar2	12	1	3	11	0	0	1	2	1
blocks1	15	5	4	10	0	2	5	1	0

time. The second clustering stage iterates over the pairwise clusters to form the final wing clusters. A new wing cluster is formed at the completion of each iteration which has a time complexity of no more than $O(\binom{n}{2})$. Note that with n wing samples, there can be as many as n wing clusters. Thus, the wing clustering module has a worst case time complexity of

$$O(n^2) + O(\binom{n}{2}) O(n) = O(n^3).$$

Wing Extensions and Intersections

The wing extensions and intersections module in POLY-3 is similar to that of POLY-1 and POLY-2. Assume that there are m different wing clusters, C_1, C_2, \dots, C_m with n_1, n_2, \dots, n_m wings, respectively, where $n_1 + n_2 + \dots + n_m = 2n$. Following the same analysis as in Section 4.6, the time complexity of the wing extension and intersection step over all wing clusters is bounded above by $O(n^2)$.

Resolving Undecided-Segments

The module for resolving the undecided-segments in POLY-3 is similar to that of POLY-2. They only differ in the rules used to form must- and unwanted-segments. So the worst case time complexity can be expected to be the same. As in POLY-2, the algorithm iterates over the set of remaining undecided segments until either no more undecided-segments remain or no more updating is possible. The worst case scenario in resolving all the undecided-segments is when only one undecided-segment is updated during each iteration. Note that once a segment is "decided" it can never be changed back to the undecided mode. Thus for wing cluster C_i , which has at most n_i^2 undecided-segments, the worst case time complexity is

$$O(\sum_{j=1}^{n_i^2} j) = O(\frac{(1 + n_i^2)(n_i^2)}{2}) = O(n_i^4).$$

Over all wing clusters, the total time complexity is

$$\sum_{i=1}^m O(n_i^4),$$

which is bounded above by

$$O((n_1 + n_2 + \dots + n_m)^4) = O((2n)^4) = O(n^4).$$

Merging of Planar Faces

The analysis of the merging module in POLY-3 is more complex than in POLY-2 due to the existence of maybe-segments. Given a wing cluster C_i with n_i wings in the cluster, at the completion of the reconstruction the numbers of must- and maybe-segments are bounded above by n_i and $2n_i$, respectively. Thus wing cluster C_i would contribute at most $3n_i$ segments to be merged with segments from other wing clusters. Summing over all wing clusters, there would be at most

$$\sum_{i=1}^m 3n_i = 3 \sum_{i=1}^m n_i = 3n$$

must- and/or maybe-segments to be merged. Pairwise merging those segments yields the line segments in the final LDG. This merging step has a time complexity of

$$O\left(\binom{3n}{2}\right) = O(n^2)$$

Note that resolution of must/maybe label on the line segments of the LDG is also done during this merging step.

As indicated in Section 5.3.5, with the inherent round-off errors of floating point computations, a true junction in the LDG may be represented by many scattered, yet close-by, junctions. A junction clustering step is performed to group those multi-

junctions into one representative junction. Given the number of line segments to be merged is $3n$, the maximum number of junctions that can arise is $O(\binom{3n}{2}) = O(n^2)$. Using the pairwise clustering strategy as for merging line segments, the junction clustering step requires has a theoretical time complexity of

$$O(\binom{n^2}{2}) \cdot O(\binom{3n}{2}) = O(n^6).$$

However, in practice, most of the line segments do not intersect and thus do not produce any junctions. The worst case time complexity of $O(n^6)$ should be treated as a theoretical upper bound.

Thus, the total time required for merging the faces is bounded by

$$O(n^2) + O(n^6) = O(n^6).$$

Overall Worst Case Time Complexity

Given the above complexity analysis of individual modules, the algorithm POLY-3 has the worst case time complexity of

$$O(n^3) + O(n^2) + O(n^4) + O(n^6) = O(n^6).$$

5.6 Summary

In this chapter, we presented a third algorithm, POLY-3, for reconstructing the LLDG of origami/polygonal scenes. POLY-3 differs from its predecessor in many ways. For one, it is constructed to work with wing samples detected from real images. As such, it allows for an imperfect set of wing samples as input. Missing and spurious wings, disallowed in POLY-1 and POLY-2, are allowed with POLY-3. Furthermore, the unrealistic assumption of infinite computation and measurement accuracy are

dropped. POLY-3 is an algorithm designed to work in the real, yet imperfect world. In place of those idealistic assumptions is a set of 6 heuristic rules that reason about junctions and regions to inject must-segments and to reject unwanted-segments. A junction catalogue is not used and the original range, intensity or fused images are also not needed.

The input set of possibly imperfect wing samples as detected from the wing detector algorithm in Chapter 3 is preprocessed to merge overlapping compatible wings and to remove any spurious wings. The all important step of clustering wings into planar group was shown to be realizable given the accuracy of our wing detector. Heuristic rules are used to reconstruct the LDG of each planar group of wings. Some faces may not be completely reconstructed due to missing wings. Merging of different faces from different planar groups requires that a must-segment in the final LDG be substantiated by the two plane groups that it is in. This conservative approach guards against introducing spurious segment in the final LDG. By trying to be sure that no untrue must-segment exists in the reconstructed LLDG, a higher level object recognition module can proceed without wasting time second guessing the validity of the line feature.

The algorithm has been implemented and tested on the same test cases of POLY-2 to ensure that its performance is comparable to that of POLY-2 when a complete set of wing is present. POLY-3 has also been tested on a number of imperfect sets of wings as acquired from real imagery through our wing detector. The results show that reconstruction of partial LLDGs is tenable given the accuracy of our wing detector and the set of heuristic rules.

Finally we note that POLY-3 is only as good as the wing detector will allow it to be. If the accuracy of the detected wing falls outside of the fixed tolerance range for error, then the clustering step may fail (wings put in the wrong group) causing many segments in the true LDG to be missed. Future research will aim at

improving the robustness of the algorithm to tolerate a wider range of wing detection inaccuracies. Also we note that similar to POLY-1 and POLY-2, POLY-3 is suitable for parallel implementation. Once the wings are clustered, reconstruction can proceed independently. As with POLY-2, a parallel version of the algorithm that allows for different planar groups to communicate during reconstruction may facilitate the undecided-segment decision process.

CHAPTER 6

Reconstruction of the LLDG - Quadric Surface Scenes

6.1 Introduction

As with line drawing analysis for polygonal scenes, some attempts have been made to understand the line drawings of scenes of limited classes of curved objects [25, 83, 92, 91, 93], but no one has been successful in building a line extractor to demonstrate that the two step approach to scene interpretation (line extraction followed by line labeling/interpretation) is viable. To our knowledge Trytten [119] is the only other researcher who is working toward a labeled curved line drawing graph by extracting and labeling line segments simultaneously.

In this chapter, we will assume that objects in the scene are either δ -quadrics or objects with δ -quadric faces, and that the scene can be composed of multiple objects which may occlude one another. In the following section, we will argue that under ideal wing sensing and infinite computation accuracy assumptions, the POLY-1 and POLY-2 algorithms of Chapter 4 can be modified to reconstruct the LLDG of scenes in the quadric domain. In Section 6.3, we drop the idealistic assumptions about the sensor and computational accuracy and point out two major obstacles in our

approach to reconstruction. A new algorithm will be proposed to work around one of the difficulties. The solution to the second problem is left for future research. Examples of reconstruction by both algorithms are provided.

6.2 Perfect Reconstruction Algorithm

Initially, reconstruction of the LLDG from wings seems to be made more difficult by the expansion of the object domain to include objects with δ -quadric surfaces. However, under ideal assumptions, reconstruction in this domain is no more difficult than reconstruction in the polygonal domain. The POLY-1 and POLY-2 algorithms depicted in Chapter 4 can be readily adapted to work in this expanded domain. In fact, the only modifications are the wing clustering and the wing extension steps. In the following subsections, we will reiterate the idealistic assumptions, point out the necessary modifications to the previous algorithms and present some examples of reconstructed scenes.

6.2.1 Wing Sensing Assumptions

Recalling from Section 4.3, the ideal wing sensing and computation accuracy assumptions are:

Wing Detector Assumptions:

- A₃) The view is non-accidental and the “wing sensor” produces at least one “correct” wing sample that projects onto a line segment, curved or straight, in the LDG for each segment that would be visible in the LDG. (Recall the definition of a non-accidental view from Section 2.3.)
- A₄) There are no spurious wings.

Computation Accuracy Assumption:

A_5) Measurements and computations are infinitely accurate.

Note that the no missing wing assumption (Assumption A_3) has been modified to account for the difficulty in defining edges of curved objects in 3D. Together with the assumption on computational accuracy, the two assumptions guarantee that every visible line segment, curved or straight, of the LDG is represented by at least one physically "correct" wing sample (meaning that the wing surface label contains the correct quadric shape and parameters for the adjacent regions) and that co-quadric wings have a common wing surface label. Two wings are co-quadric if they lie on the same quadric surface in 3D.

6.2.2 Wing Clustering

The first step in the reconstruction process is to cluster the wings into "common" surface groups. Previously, all surfaces were planar and the wings were grouped according to the planar equations of the wing surfaces. With the addition of quadric surface types, this clustering step is broken into a two-step process. Recall that the wing surface label contains knowledge of both the qualitative surface shape and the quantitative surface shape parameters. Using the qualitative shape information, the wings are first clustered into *planar*, *circular*, *cylindrical* and *conical* groups. Each group is then further clustered by using the shape parameters. The second grouping is only possible if one is confident about the accuracy of the recovered parameters but this is guaranteed by the computation accuracy assumption.

6.2.3 Extension of Wings

Upon grouping of wings into clusters, reconstruction of individual wing clusters proceeds independently. As in POLY-1 and POLY-2, the first step toward this recon-

struction is to form a graph embedding of the true LDG. This was accomplished by extending the straight wing contours and forming the intersections as junctions in the polygonal scenes. Extending curved wing contours is much the same. For the quadric surface domain, we know that line segments in the LDG must be of type linear, circular or elliptic (see Section 2.4). Given our computation accuracy assumption, coupled with the wing contour information accompanying each wing sample, the wing contour can be extended without error. Extending the curved wing contour in practice, however, is a much more difficult task. In Section 6.3, a possible remedy to this problem will be offered.

6.2.4 Reconstruction of Individual Wing Group

The extended wing contours intersect to form the initial set of undecided-segments. As before, those segments that overlap wing contours have to be must-segments (Assumptions A_3 and A_4). In the POLY-1 algorithm, range data on both sides of an undecided-segment are re-sampled to justify its presence as a must-segment or to reject it as an unwanted-segment. If the range data is available for re-sampling, then the same procedure can be applied to determine all the must-segments in the graph.

Rather than resampling, the POLY-2 algorithm rebuilds the LDG based on geometric constraints derived from the physical structure of the LDG. Notice that the constraints used in deriving the reconstruction rules all still hold under the new object domain. To use Rules 4.1 and 4.2, the junctions in the LDG of individual quadric surfaces must all have even degree. This condition is satisfied by the definition of δ -quadrics. The constraints leading up to the development of Rule 4.3 are stated in Theorem 4.3 and Corollary 4.3.1, which already are defined in terms of δ -quadrics. Finally, the no missing wing assumption (Assumption A_3) gives us Rule 4.4. The four rules are now restated with proper choice of words to reflect the possibility of curved segment types in the LDG.

Rule 6.1 An undecided-segment seg_{ij} can be discarded if $deg_{must}(v_i)$ is even and $deg_{und}(v_i) = 1$; or if $deg_{must}(v_j)$ is even and $deg_{und}(v_j) = 1$

Rule 6.2 An undecided-segment seg_{ij} is a must-segment if $deg_{must}(v_i)$ is odd and $deg_{und}(v_i) = 1$; or if $deg_{must}(v_j)$ is odd and $deg_{und}(v_j) = 1$.

Rule 6.3a Given two adjacent line segments $Mseg_{ij}$ and $Useg_{il}$ where $Mseg_{ij}$ is a must-segment and $Useg_{il} = SegCCW_{v_i}(Mseg_{ij})$ is an undecided-segment. Assume that the wing group being processed has the surface representation P_δ . Then $Useg_{il}$ can be discarded as an unwanted-segment if

$$RL-CCW_{v_i}(Mseg_{ij}) = \{P_\delta\} \text{ and } P_\delta \notin RL-CW_{v_i}(Useg_{il})$$

or

$$RL-CCW_{v_i}(Mseg_{ij}) \neq \{P_\delta\} \text{ and } RL-CW_{v_i}(Useg_{il}) = \{P_\delta\}.$$

Rule 6.3b Given two adjacent line segments $Mseg_{ij}$ and $Useg_{il}$ where $Mseg_{ij}$ is a must-segment and $Useg_{il} = SegCW_{v_i}(Mseg_{ij})$ is an undecided-segment. Assume that the wing group being processed has the surface representation P_δ . Then $Useg_{il}$ can be discarded as an unwanted-segment if

$$RL-CW_{v_i}(Mseg_{ij}) = \{P_\delta\} \text{ and } P_\delta \notin RL-CCW_{v_i}(Useg_{il})$$

or

$$RL-CW_{v_i}(Mseg_{ij}) \neq \{P_\delta\} \text{ and } RL-CCW_{v_i}(Useg_{il}) = \{P_\delta\}.$$

Rule 6.4 Given a must-segment $Mseg_{ij}$ and an undecided-segment $Useg_{ik}$ where $Useg_{ik} = SegCW_{v_i}(Mseg_{ij})$ or $SegCCW_{v_i}(Mseg_{ij})$. If $Mseg_{ij}$ and $Useg_{ik}$ are not co-curvilinear and $WingDir_{v_i}(Useg_{ik}) = \text{False}$, then all (undecided) segments co-curvilinear to and on the side of $Useg_{ik}$ with respect to v_i including $Useg_{ik}$ can be deleted.

The resolution of the undecided-segments is solely based on reasoning about each junction and region labels, and makes no use of the physical properties (such as being

planar or otherwise) of the projecting surface. Thus, it is not surprising to see that those rules also apply in the reconstruction of the LDG in the quadric domain.

6.2.5 Quadric LLDG Reconstruction Algorithm: QUAD-1

The reconstruction algorithm given below is essentially the same as the algorithm for reconstruction in the polygonal domain except for the different choice of words to account for curved segments in the LDG. Examples of reconstructed LLDGs from complete wing samples are depicted in Figure 6.1. In hand-tracing those examples, we observed that it is less likely for curved surfaces to be co-quadric (as compared to planar surfaces to be coplanar); consequently, few wings would be clustered into co-quadric group, making the reconstruction of faces in each wing cluster a simpler task. All of the examples in Chapter 4 where only planar faces are allowed are all hand-traced and reconstructed by this algorithm.

- (1) Cluster wing segments by quadric equation
- (2) For each quadric group
 - (2.1) generate a curve for each wing
 - (2.2) compute the intersections among all curves
 - (2.3) preserve line segments which overlap a wing
 - (2.4) remove or preserve line segments by using **Rules 6.1, 6.2, 6.3 and 6.4.**
 - (2.4.1) if $V_{E1} \neq \text{null}$, apply **Rule 6.1** then goto (2.4.1)
 - (2.4.2) if $V_{O1} \neq \text{null}$, apply **Rule 6.2** then goto (2.4.1)
 - (2.4.2) apply **Rules 6.3 and 6.4** to a junction v_i of V_{und}
 - (2.4.3) if $V_{und} = \text{null}$ or no more changes are possible then goto (2.5)
else goto (2.4.1)
 - (2.5) delete redundant intersections
- (3) Merge LDGs of individual quadric faces
- (4) Compute 3D vertices by intersecting curve equations
- (5) Retain all line segment labels from wings
- (6) Output LLDG with junctions, lines, and regions all labeled.

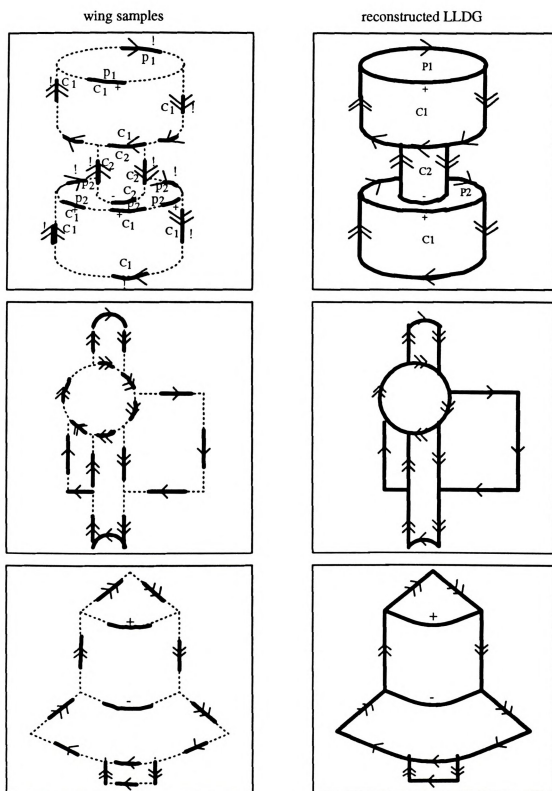
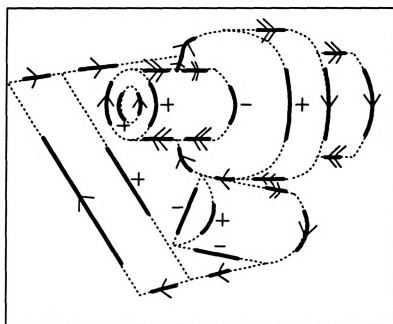


Figure 4.14. cont. (see page 179 for caption)

wing samples



reconstructed LLDG

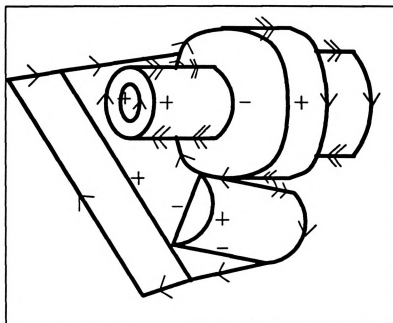


Figure 6.1. Examples of reconstruction of LLDG from wing samples of quadric scenes.

6.3 Heuristic Reconstruction Algorithm

6.3.1 Wing Clustering

The first step of all LDG reconstruction algorithms that have been presented is the clustering of wing samples. This clustering step is non-trivial in practice. There are many pitfalls which make wing clustering difficult. Recall that the wing surface label contains the qualitative surface type and the parameter estimates of the assumed surface model. If the wing detector wrongly classifies the surface patch then there is no hope of correctly clustering that wing into the “correct” wing cluster. Even if the “correct” surface model were assumed, if the surface parameters cannot be recovered accurately then wings may be placed in the wrong cluster. Worse yet, each wing could form cluster of its own if the error in estimation is large.

Fortunately, we have been able to escape such a predicament so far. In the POLY-1, POLY-2 and QUAD-1 algorithms, the measurement and computation accuracy assumptions ensure that all coplanar and co-quadric wings have the same wing surface labels; thus they can be clustered easily without mistake. In POLY-3, we relied on the fact that the wing detector is able to estimate the planar parameters accurately to correctly cluster the wings into proper groups. With the addition of quadric surfaces, the problem is not so easy to solve. In Chapter 3, we have concluded that the wing detector is able to classify qualitative surface shape more accurately than to estimate the quantitative shape parameters. Furthermore, parameter estimates from two far apart regions of a quadric surface could deviate by an unacceptable margin.

Given the difficulty in wing clustering, one has to ask: “Can this step be bypassed in LDG reconstruction?” Perhaps, reconstruction of the LDG can be achieved without having to cluster wings at all. In this section, we will present a heuristic algorithm which does not cluster wings; rather, the entire LLDG is to be reconstructed from the entire set of sampled wings at once.

6.3.2 Wing Extension

Extension of wing contours across the entire imaging (bounded) plane is also an essential part of all of the LDG reconstruction algorithms. The accurately extended segments together with their intersections form a graph of segments and junctions that is a superset of all segments and junctions of the LDG. Under ideal assumptions, extending wing contours is simple. Extending straight wings detected from real images of the polygonal domain has also been shown to be realizable given the accuracy of our wing detector. However, the same cannot be said about extension of curved wing contours. As noted in Chapter 3, the wing contour descriptor, which is derived from fitting a 2D curve model to selected edge points within each subimage, is only a *local* curve approximation; it may not reflect the true shape of the global curve. Blindly extending each wing contour may result in many supposedly co-curvilinear curves that are potentially damaging to the heuristic reconstruction of the LDG.

One possible solution to this dilemma is to use the technique of the Hough Transform (HT) [58], popularized by Duda and Hart [32], to accumulate evidence about existence of different curves and their locations in the image. The main advantages of the Hough Transform for curve detection are that little has to be known about the location of the curve as long as the shape of the curve can be described parametrically; furthermore, it is relatively unaffected by gaps in curves and by noise [7]. In [6], Ballard presented a generalized Hough Transform technique for detection of arbitrarily shaped curves that cannot be easily parameterized. The biggest drawback with the HT techniques are that they generally are very time and memory consuming and are therefore expensive for practical computer vision tasks. For this reason, much research on the HT has aimed at reducing the storage (parameter) space and/or speeding up the process (*e.g.*, [3, 64, 79]). Others have concentrated on detection of

certain curve shape [23, 88] or surface shape [89, 109]. A survey of different variations of Hough transform can be found in [65].

Hough transform techniques have also been used in the object recognition aspect of computer vision. Object recognition and pose identification via transformation clustering is a Hough based technique [75, 114]. However, recent investigation by Grimson and Huttenlocher on the sensitivity of using the Hough Transform to perform transformation clustering suggests that under moderate levels of noise in sensory data, occlusion and image clutter, the HT may generate many false solutions [46].

Part of our future research plan is the study of Hough based techniques for grouping co-curvilinear wing contours and to recover the underlying curve from those wing contours.

6.3.3 A Heuristic Approach

The purpose of clustering wings in all of the previous algorithms was to weed out unnecessary wings and to keep only the co-quadric wings for quadric surface reconstruction. However, if a wing is wrongly clustered, it, in effect, has introduced a spurious wing in one cluster and its rightful cluster will be missing a wing. The effect is potentially damaging if several wings are misclustered. Rather than risking the damaging effect of wrongly clustered wings, we shall propose an algorithm that uses the entire set of wing samples to reconstruct the LDG at once. The surface equations of the wing surfaces used for wing clustering are only used in this algorithm to verify the hypothesis that two line segments should be merged. Figure 6.4 provides an accompanying example for the explanation of the algorithm below.

Step 1: Extension of Wings

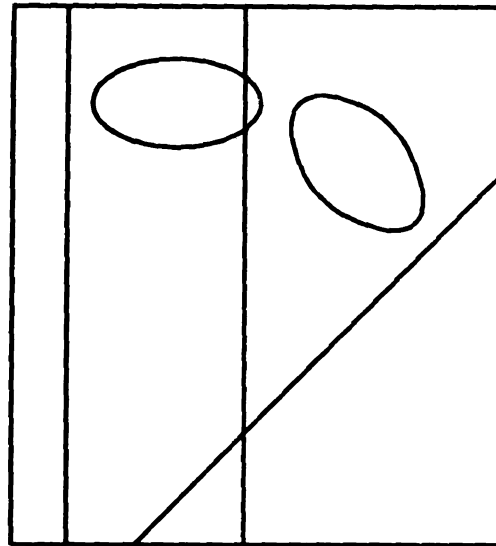
Although we have side-stepped the issue of wing clustering, our strategy still revolves around creating a graph G in which the true LLDG is embedded, and to

remove unwanted-segments and verify must-segments using heuristic rules. As discussed above, extension of wings is also a difficult task made worse by inexact shape representation of the wing contour. Although our wing detector can reliably differentiate surface shapes, the wing contour shape can sometimes be misrepresented especially under the presence of noise. The accuracy of wing segment shape representation depends strongly on the window size used in detecting that wing. For example, with a small window, due to shortness of the edge image in the window, many wings may be detected as being straight. Future research will aim at better estimation of the wing segment shape.

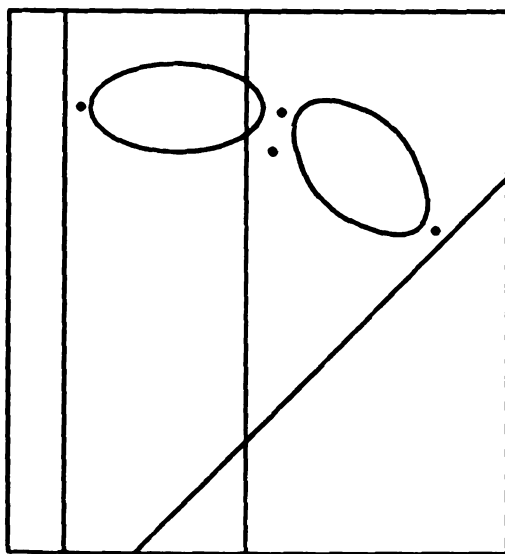
Step 2: Determination of Junctions

Assuming that wings can be reliably extended, some housekeeping jobs are required before starting the reconstruction loop. First the wing labels are propagated to the entire extensions. Then the intersections of those wing extensions must be resolved. Under an ideal situation, only one junction will arise when many line segments meet at supposedly one junction. In practice, supposedly co-terminating segments may intersect at more than one point or may fail to intersect at all (see Figure 6.4 (b)).

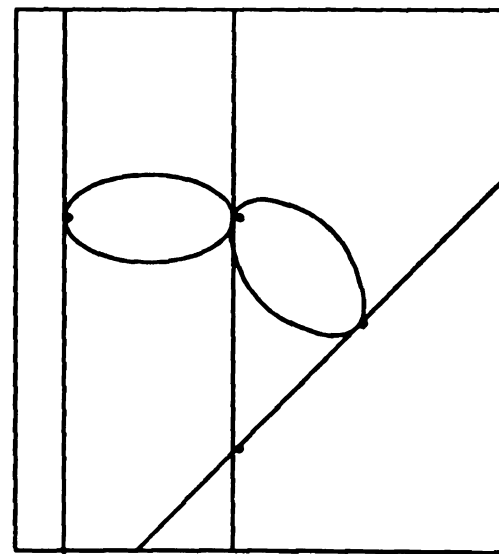
To find missing junctions, if the minimum Euclidean distance d between two non-intersecting line segments is less than some threshold γ , then the two segments are adjusted to meet along the shortest path between them. An example is provided in Figure 6.2 (a). Note that already intersecting segments are exempt from this procedure. Although intersecting wing extensions and merging of nearby segments may create many split junctions, the junctions are clustered so that nearby junctions are pulled together into one unique representative junction (see Figure 6.2 (b)). Finally, the endpoints of all wing segments are also denoted as junctions.



wings fully extended



(a)



(b)

Figure 6.2. Determining set of all possible junctions in the LDG. (a) Nearby non-intersecting segments are merged at •. (b) Nearby scattered junctions are clustered into one.

Step 3: Initial Classification of All Segments

Initially all the segments in G are denoted as undecided-segments. Update the segment labels in G by labeling all wing segments as must-segments. An undecided-segment is a maybe-segment if it is coincident and co-curvilinear with exactly one must-segment at one of its two endpoints. The updated G is shown in Figure 6.4 (b) where undecided-segments are represented by broken lines, maybe-segments by solid lines and must-segments by thick solid lines.

Step 4.1: Removal of Unwanted-Segments

We are now ready to invoke the heuristic rules to remove any undecided-segments as unwanted-segments. Recall that each region of the LDG can only correspond to a single face of the object in 3D. Therefore, theoretically, each region should only have one label. In reality, computation inaccuracy prevents the segments surrounding that region to have the same exact label (remember that the wing labels are propagated to all its extensions in Step 2). Also recall that the wing surface label consists of both the surface type and the parameters of that surface. Our wing detection experiments suggest that although the recovered surface shape parameters may vary throughout the surface patch, the qualitative surface type can be accurately predicted. For convenience, denote that surface type as the coarse label and the actual parameters as the fine label of the wing surface label.

The following rule checks for inconsistency of the coarse labels of two adjacent segments on the region bounded by the two segments. If they differ and one is an undecided-segment while the other is a maybe- or must-segment, then the undecided-segment is discarded. No action is taken if both segments are undecided-segments because we have no knowledge of which segment is likely to be unwanted. The rule is graphically illustrated in Figure 6.3 (a).

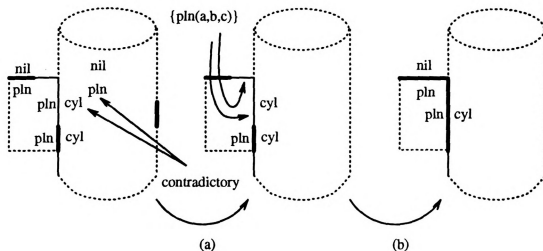


Figure 6.3. Graphical illustration of the heuristic rules for reconstructing LDG of curved object scenes. (a) Rule 1, removal of undecided-segments. (b) Rule 2, rectification of must-segments.

Rule 6.5 *An undecided-segment seg_{ij} can be discarded as an unwanted-segment if seg_{ik} is an adjacent, maybe- or must-segment and the coarse label of the bounded region of the two segments are different.*

Removal of undecided-segments may leave another undecided-segment dangling by itself (i.e., not attached to any other segments). Therefore, any dangling undecided-segments are also removed as unwanted-segments. In addition, phantom junctions, those having no segments attached or incident to two co-curvilinear segments, are removed from G , and the list of maybe-segments is updated accordingly.

The above process is applied repeatedly until no more undecided-segments can be removed. Many of the undecided-segments in the running examples are removed by the above procedure and maybe-segments are extended as shown in Figure 6.4 (c).

Step 4.2: Verification of Must-Segments

While coarse labels are used to check for inconsistency of region labeling by two adjacent segments, they can also be used to check for consistency.

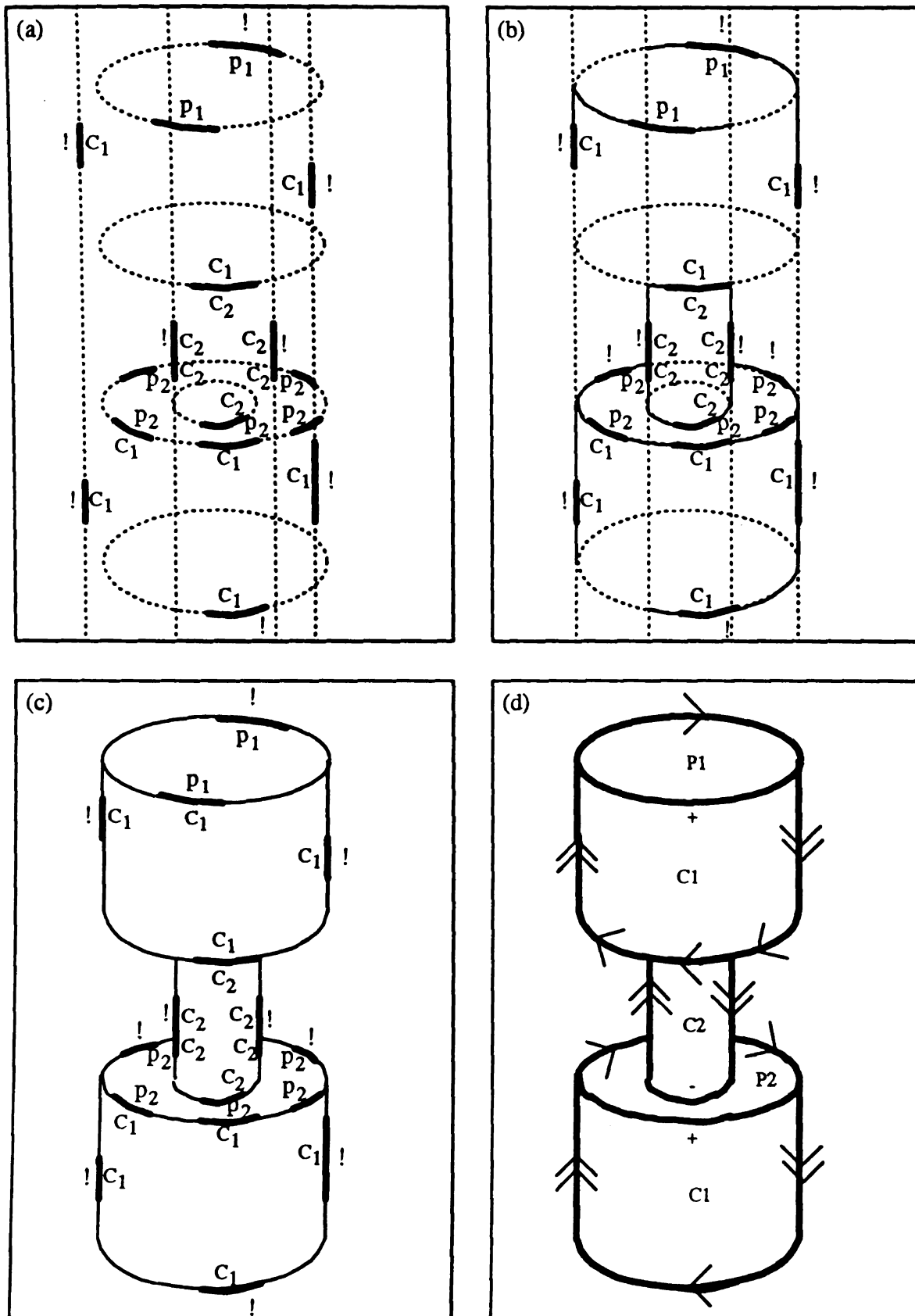


Figure 6.4. Reconstruction of LDG of a curved surface scene. Broken lines are undecided-segments, solid lines are maybe-segments and thick solid lines are must-segments). (a) Wing samples and their extensions. (b) After junction detection and clustering. (c) Result of repeat application of Rule 6.5. (d) Result of one application of Rule 6.6.

If the region labels are consistent, and both segments are maybe-segments or one is a must-segment while the other is a maybe-segment, then it is hypothesized that both segments are must-segments. The hypothesis is verified or rejected by comparing the finer label (parameter estimates of the projecting surface) of the two segments. If the estimated parameters are reasonably close, then the hypothesis is accepted; otherwise, rejected. This hypothesis verification step is similar in principle to the wing clustering step that we have cited as being too unreliable and consequently abandoned. However, there is a major difference in these two approaches. Wing clustering was performed over all sampled wings; wings that are far apart, though on the same surface, need to be clustered into the same group. Here, only two sets of parameters are compared, and they are propagated from two wing segments that fall on adjacent line segments of the LDG, which in general means the surface patches from which those wings are derived are close in 3D. Again, from the wing detection experiments, we have concluded that nearby wings have more consistent parameter estimates than far apart wings of the same surface, except in the planar case where there is not much difference. Therefore, hypothesis verification by comparing the parameter estimates is reasonable.

Rule 6.6 *Two adjacent segments seg_{ij} , seg_{ik} should both be must-segments if (1) both are must- or maybe-segments or one of each; (2) the coarse labeling of their bounded region is consistent; and (3) the fine labeling of their bounded region is also consistent.*

The result of applying this rule to the the running examples is depicted in Figure 6.4 (d). Again, any phantom junction resulting from this operation is removed.

Step 5: The Final LLDG

By repeated application of the above two steps, we arrive at the final LLDG as in Figure 6.4 (e). It only remains to label the regions with a unique label. Note that

any closed region in the LDG must have a consistent coarse label on all surrounding segments (Rule 6.6). If the original range image is available for resampling, one could refit the surface model (coarse label of the region) to points sampled from within this region on the range image. If the range image is not available, then depending on the application, all the estimates or perhaps the mean of all estimates could be passed onto the next higher level of recognition module.

6.3.4 Heuristic Quadric LLDG Reconstruction Algorithm: QUAD-2

The complete heuristic algorithm is outlined below.

- (1) Extend each wing segment (no clustering)
- (2) Form junctions by computing the intersections among all curves and propagate the wing labels of each wing segment to its extensions; denote this graph as G
- (3) Initial classification of must-segments, maybe-segments, and undecided segments
- (4) Remove or preserve maybe-segments by heuristic Rules 6.5 and 6.6
 - (4.1) Repeatedly apply heuristic Rule 6.5 to all remaining undecided-segments in G until no more undecided-segments can be thrown out as unwanted-segments
 - (4.2) Apply heuristic Rule 6.6 to all remaining maybe-segments in G to rectify any must-segment
 - (4.3) If G is updated, loop back to step (4.1)
- (5) Resolve region labels and output (partial) LLDG.

6.3.5 Examples

Two examples involving missing wings are given in Figure 6.5. In Figure 6.5 (a), we removed 5 wings from Figure 6.4 (a) that we thought would be less likely to be sensed from real fused image. Using algorithm QUAD-2, the reconstructed LLDG revealed

the two cylinders at the end. However, one curve segment that does not belong in the true LLDG was inferred. In Figure 6.5 (b), two more wings were removed which results in just one big cylinder and two non-existing limb contour segments in the reconstructed LLDG. In the defense of the algorithm, we must point out that the most revealing features about the object in the scene, the curvature-L junctions and the three-tangent junctions, were detected; furthermore, the reconstructed LLDGs mimic how the human visual system would have reconstructed the scenes without given prior knowledge about the objects in the scenes.

In Figure 6.6, more interesting facts about the algorithms are revealed. In Figure 6.6 (a), the complete LLDG are reconstructed despite that fact that there are three missing wings. In Figure 6.6 (b), a missing wing caused 4 maybe-segments not to be resolved, which indicates that the reconstructed LLDG is not complete. In Figure 6.6 (c), a complete set of wing samples are supplied for reconstruction; however, 4 maybe- and 3 undecided-segments are not resolved by the reconstruction algorithm. Those examples show that the reconstruction rules used in algorithm QUAD-2 are not sufficient. Better decision rules await to be discovered to prevent situations like that of Figure 6.6 (c) from occurring.

6.4 Summary

Extending reconstruction techniques to the domain of quadric-surface objects proves to be quite challenging. In this chapter, we have shown that under the same set of idealistic assumptions about the input wing set and the computation accuracy as in Chapter 4, the LLDG reconstruction algorithms of POLY-1 and POLY-2 can be directly extended to reconstruct the LLDG in the expanded domain. The extension is possible because the algorithms made no assumptions about the linearity of the line segment and the planarity of the surfaces. Reconstruction was performed based purely

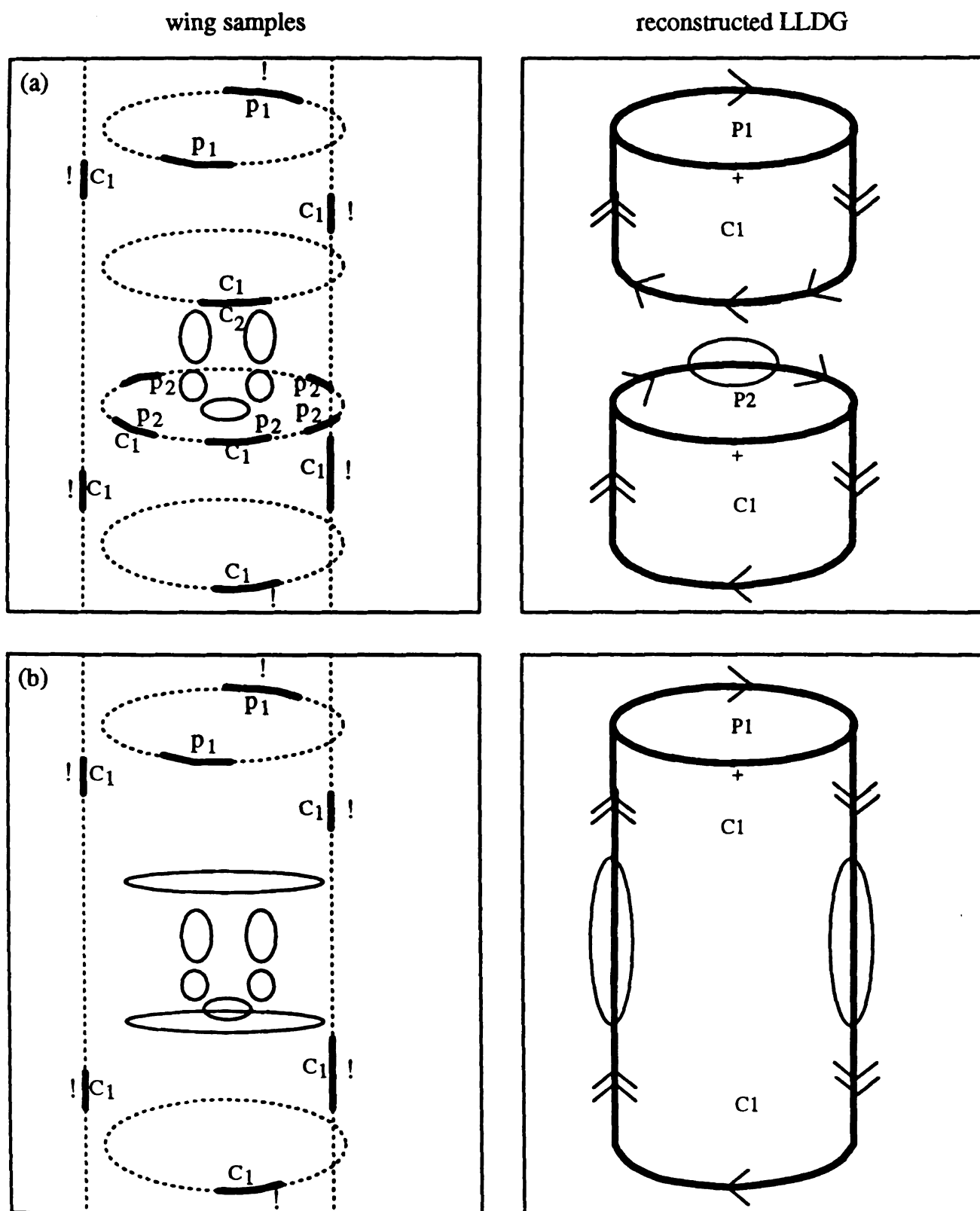


Figure 6.5. Reconstruction of LLDGs from imperfect wing samples using heuristic algorithm QUAD-2.

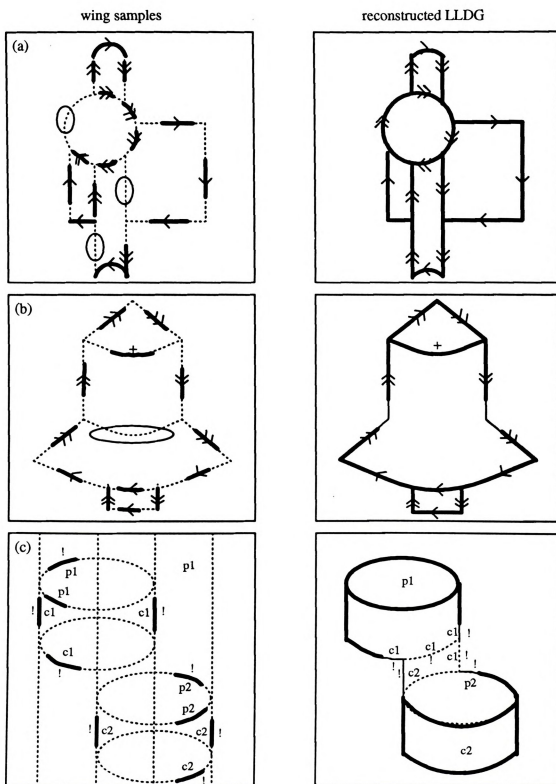


Figure 6.6. More LLDGs that are reconstructed by the heuristic algorithm QUAD-2.

on the geometrical constraints of the underlying line drawing graph. A modified algorithm, QUAD-1, was outlined.

A couple of challenging problems arise when the idealistic assumptions are dropped. First, the wing clustering step, which was successful for POLY-3, will be hard to duplicate given our knowledge about the quality of the quadric surface wings detectable by our wing finder. We argued that wing clustering requires more accuracy on the estimated surface shape parameters than currently available. Instead of risking placing wings in the wrong cluster, a new algorithm, QUAD-2, is proposed to bypass the whole clustering step and to reconstruct the LLDG at once from the set of input wings. The second problem when dealing with realistic situations, has to do with the extensibility of wing contours to form a graph embedding of the LDG. Given that wing contours only model the local image contour shape, extending it to fit the global contour is asking a lot. No solution to this problem was offered but the Hough Transform technique was suggested to be a plausible approach.

The proposed QUAD-2 algorithm has only two heuristic decision making rules. Both are based on the fact that each region in the LDG corresponds to a physical surface in 3D so that the line segments surrounding a region should all label the common region the same way. The qualitative region labels of each segment are used to discard any conflicting unwanted segments and the surface parameter estimates are checked to verify the hypothesis of a must-segment. A drawback of this algorithm is that by not clustering sampled wings into co-quadric groups and reconstructing each group independently, the transparent parallel implementation of previous algorithms is not present in QUAD-2.

Neither QUAD-1 nor QUAD-2 has been implemented. Hand-traced results were provided to exemplify the algorithms. Future research is needed to solve the difficult task of extending curved wing contours so that QUAD-2 can be aptly applied.

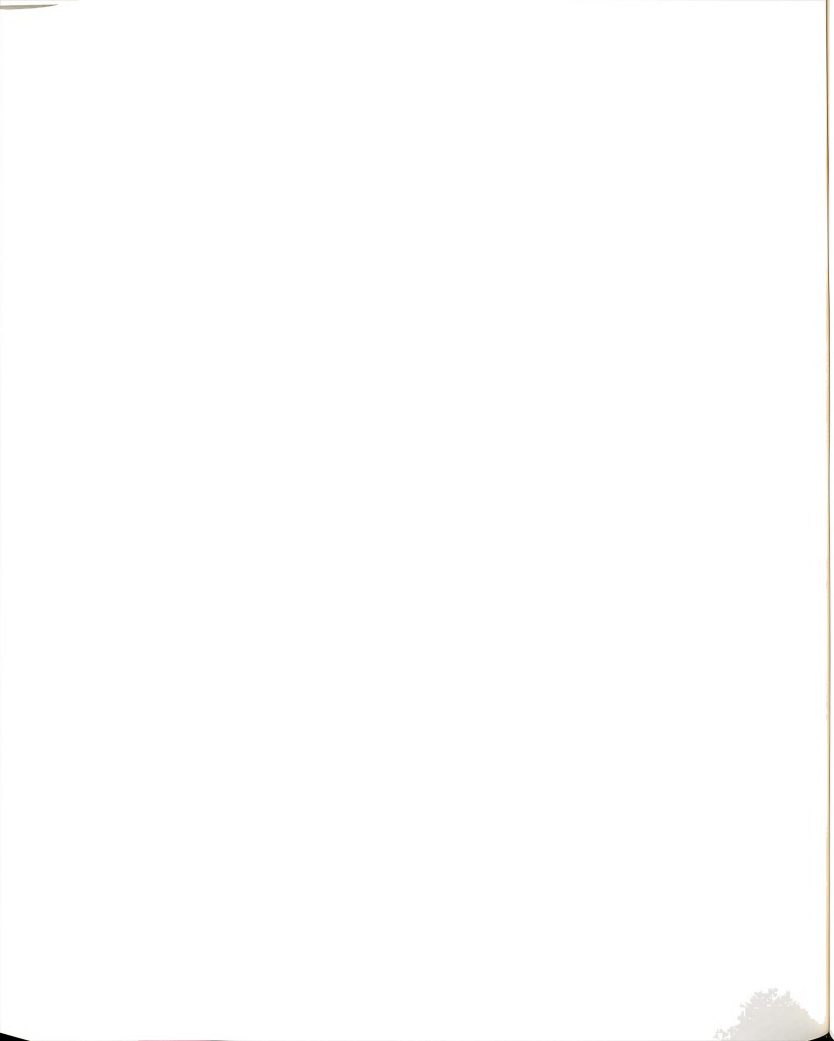
CHAPTER 7

Summary, Conclusions and Recommendations for Future Research

7.1 Summary and Conclusions

The research addressed in this dissertation dealt with extracting a labeled line drawing graph from registered range and intensity image via locally detected wing primitives. The problems of wing detection and labeled line drawing reconstruction are treated as separate issues: Wing primitives are extracted from raw fused images and reconstruction of the LLDG proceeds from the sampled wings. By making it a separate issue, limitations of wing detection will not hinder the growth of line drawing reconstruction research.

After introducing the research problem in Chapter 2, we set out to define the special terminology used in this thesis and to derive the mathematical formulation for various quadric surfaces, the image contours and image of the crease edge created by intersecting quadrics. The structure of the wing and its mathematical representation were discussed; in the process, a set of 98 wing primitives were conceived for the



quadric surface domain. It is those primitives that are to be extracted from the fused images. A labeled line drawing graph is a representation for the view of the scene with as much geometric and topological fidelity as is possible. We defined accidental views in terms of the *surface sensing property*, which dictates that each visible edge and surface must be partially sensed. Mathematical representation of various surface and contour shapes were derived and fit to facilitate the wing detection process.

The first problem that we studied was on the detection of wings from fused images. Many issues and conclusions were drawn from this study. A major contribution of this thesis is the study of the effectiveness of simultaneously fitting 3D surface data and 2D image contour data to recover surface shape and pose parameters. A large number of Monte Carlo experiments were performed and the results strongly support our hypothesis that fused fitting can better discriminate between different quadric shapes and provide more accurate estimations of the pose parameters than if surface data or contour data were used alone. The results are even more compelling when Gaussian noise of higher variance was introduced in the data. Experimentation on real images further supports this conclusion. An experiment was also conducted to study the minimum number of data points required to obtain reasonable fits. The performance of the fused and surface-only fitters stabilized when as few as 20 sample points were used for fitting. This result refutes the common belief that large number, hundreds or even thousands, of sample points are required to obtain good fits. Private communication [24] with our colleagues confirms that it is not the number of data points, rather, it is the coverage of the sample points that controls the success of the fit.

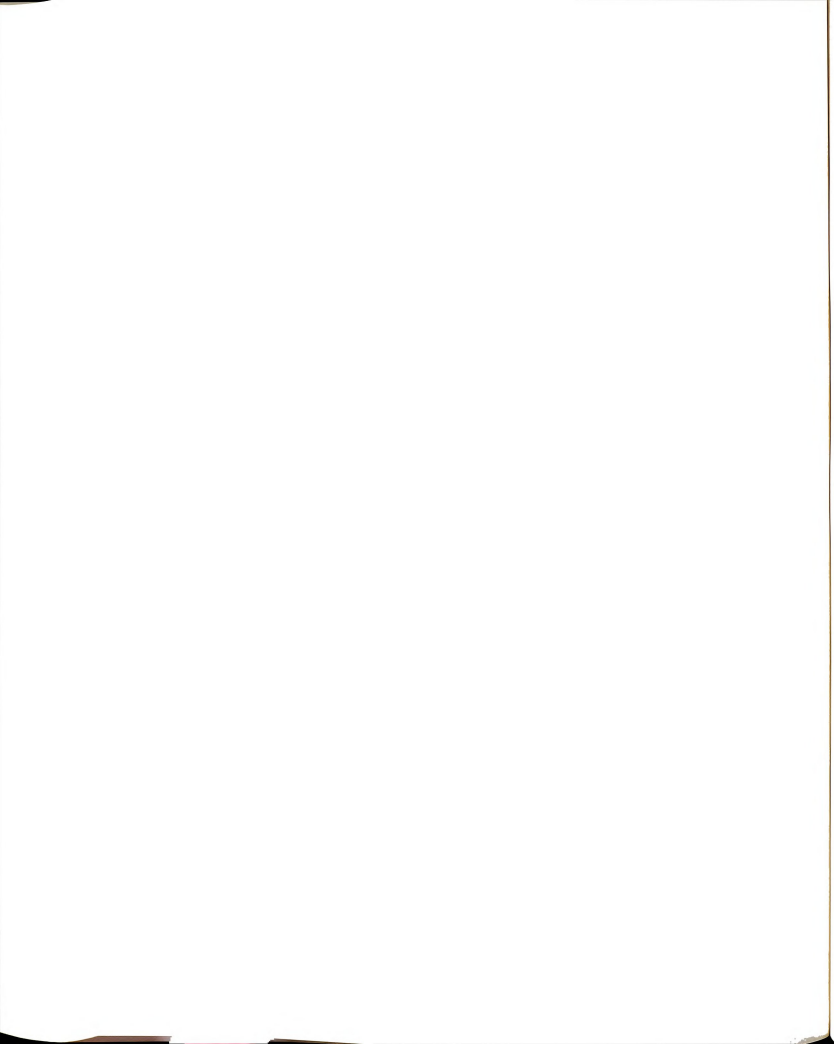
Armed with this new found fused fitting technique, we proceeded to investigate the main problem of wing detection. It is well known that edge information from objects provides strong clues about the local shape of surface regions [70] and shape-from-boundary methods have already been implemented [127]. For example, the limb

boundary of a circular cylinder should be detected as much by its straight boundary contour as by the curvature of its surface points. The elliptical contour at the lip of a cup or can is perhaps more revealing of 3D shape than is a local set of range samples as recent work has shown [41, 54]. We proposed to incorporate the limb boundary and crease edge position information to constrain and to aid the recovery of the surface shape parameters by simultaneous fitting the 2D contour information with the sampled 3D surface data points. Furthermore, the limb contour was shown to be of particular importance for it is instrumental in generating good initial parameter estimates for non-linear fitting.

The wing detection algorithm was tested on over 20 real fused images. Carefully studying the results enabled us to draw the following conclusions:

- The wings can be reliably detected in polygonal scenes. Presence of spurious wings are rare and the variation of surface parameters across a plane is low.
- The conclusion drawn from Monte Carlo experiments about the surface shape discriminatory power of fused fitting is also evident in real images. Wings near the limb boundary or crease edge contour are detected with more precision and with fewer errors in qualitative surface classification.
- Wing detection in images of quadric surface scenes reveals that the qualitative shape of the surface is often correctly recovered but the quantitative shape parameters may have large variance over the same quadric surface.

After closely examining the quality of the wing detector, we turned our attention to reconstruction of labeled line drawing graphs in Chapters 4 to 6. In Chapter 4, we studied the plausibility of reconstructing a *unique* line drawing of origami and polygonal scenes from a complete set of wing samples under some favorable conditions. The input wing sets were assumed to be complete, meaning that at least one wing was detected for each visible edge in the LDG but no spurious wings. Furthermore,

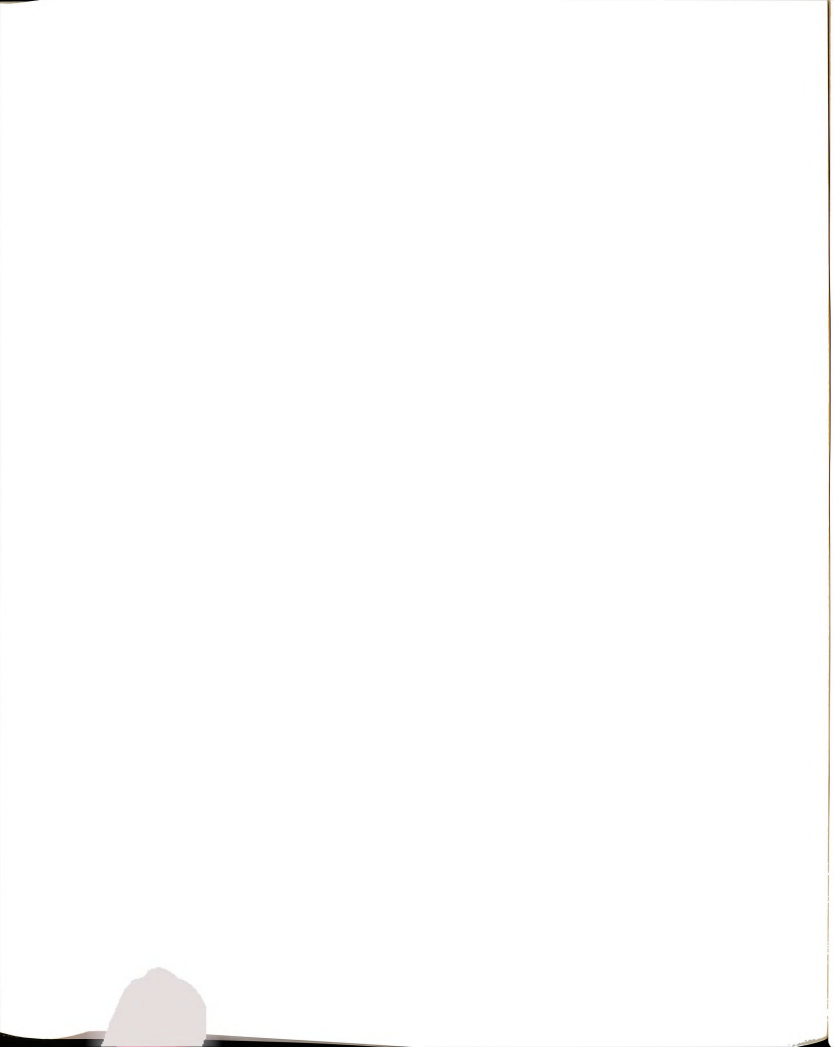


infinite computation and measurement accuracy were assumed so that the validity of the reconstruction algorithm could be accurately assessed. Under these idealistic assumptions, two deterministic algorithms, POLY-1 and POLY-2, were proposed for complete reconstruction of the LLDG. By allowing re-sampling of range image, complete reconstruction of the labeled line drawing graph is guaranteed. On the other hand, by studying the geometric constraints of the line drawing graph, rules can be deduced to facilitate the reconstruction process as in POLY-2. Since those rules are derived from the geometric properties of the LDG, they were shown to be geometrically "correct", meaning that the reconstruction rules are necessary conditions for any LDG. However, as pointed out by one of the examples, those rules are not sufficient to guarantee the recovery of a complete LLDG. The problems lies not with the reconstruction algorithm; rather, the information retained by wings about the scene may be inadequate for complete reconstruction even in an idealistic setting.

In Chapter 5, the idealistic assumptions were dropped and a new algorithm, POLY-3, was proposed to reconstruct the LLDG from the wing samples as detected by our wing sensor. The objects were still assumed to consist of planar surfaces only. Some of the problems addressed and overcome were:

1. possible inaccuracy and imprecision of wing parameters;
2. spurious wing samples that do not correspond to any physical line segment in the LDG; and
3. missing wings (*i.e.*, no wing sample for some of the line segments of the LDG).

The problem of possible inaccuracy in wing parameters was partially solved by the inherent capability of our wing sensor to produce accurate wings. However, there will always be some discrepancy in the parameters. This was overcome by introducing displacement tolerances in places where the quantitative parameter values are compared. Spurious wings could potentially cause difficult problems if they are present

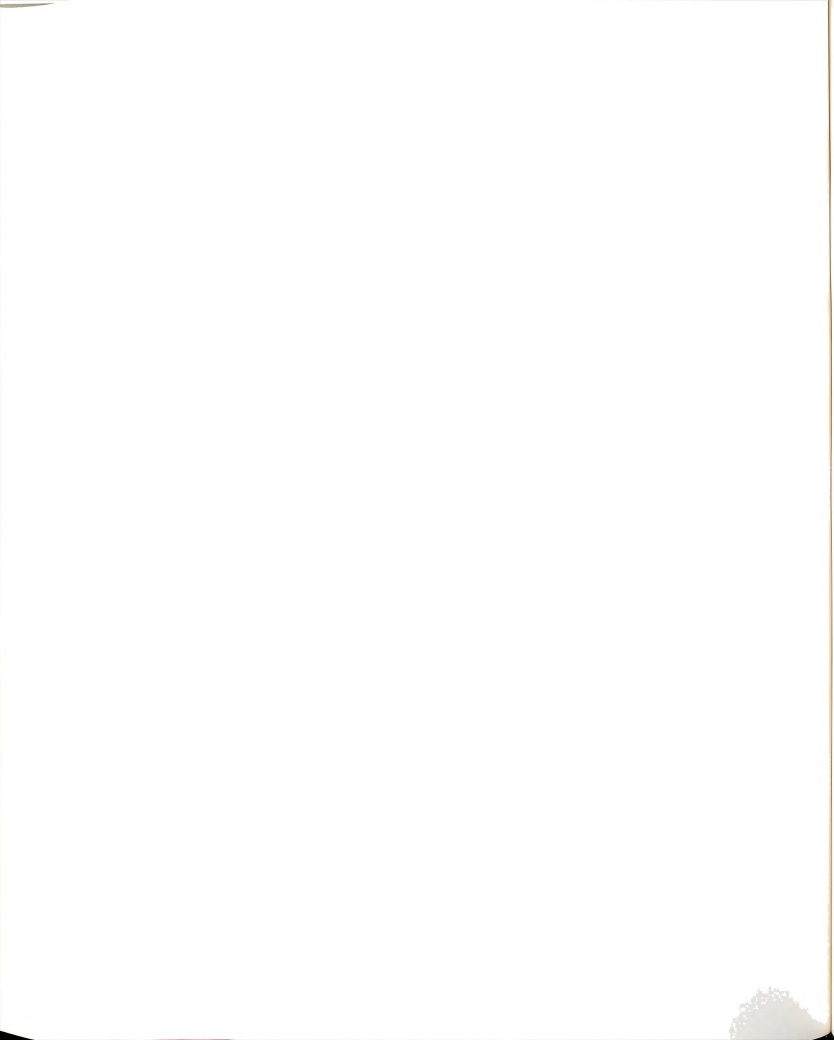


during the reconstruction. Since spurious wings do not correspond to physical edges in 3D, by checking to see if the two planar surfaces represented by the wing are actually coplanar, they can be discerned from other wings and discarded from the wing samples. Experiments have shown that spurious wings are rare and are often removed by the preprocessor. The missing wing problem is the reason why heuristic rules are used for reconstruction. By means of heuristic rules, the algorithm has no guarantee of reconstructing a complete LLDG. We conclude that the algorithm is sound by showing that (1) it is capable of generating an LLDG comparable to that of POLY-2 when a complete set of wing samples is given as input; and (2) a partial LLDG can be obtained even when the input wing set is incomplete. With this, we conclude that (partial) reconstruction of labeled line drawing graphs from fused range and intensity imagery is tenable for scenes involving origami and polygonal objects.

Having drawn such strong conclusions about planar surface scenes, we must admit that our work on reconstructing LLDG of quadric surfaced scenes is only preliminary. We argued that under idealistic assumptions, the POLY-1 and POLY-2 algorithms can be directly extended to operate in the expanded domain. Hand traced examples supported our argument. We expect that the difficulties involved in reconstructing the LLDG for the polygonal domain will only be magnified for the quadric scenes. An algorithm was offered to overcome some of these difficulties, but it has not been implemented nor fully tested. Hand traced examples are available to encourage us that we are moving in the right direction.

7.2 Contributions

The primary contribution of this research is the successful **demonstration of reconstructing a labeled line drawing graph (LLDG) from raw fused imagery.**



The wing detection algorithm and the reconstruction algorithms were tested on a large number of test cases, both synthetic and real.

A contribution is made to strengthening "wing representation theory" by the successful implementation and testing of a wing detection algorithm. In contrast to some other representational theories, we showed that the primary feature of the wing representation theory, which is the wing, can be extracted from raw fused images.

We have proposed to recover surface shape and pose parameters by simultaneously fitting 3D surface data as well as relevant 2D contour points. We have shown by both Monte Carlo experiments and by tests on real data that fused fitting is superior to either fitting surface data or contour data alone. We also found that only a small number of data points are really necessary to obtain quality fitting results. Valuable computation time is saved by sampling the data points for fitting rather than fitting all the available data points. Also within the study of wing detection, we discovered that the limb contour position information can help provide a good starting set of parameter values for the non-linear fitting routine. A good starting vector is essential for the fitter to converge and to converge to the global optimum.

The study on labeled line drawing reconstruction of polygonal scenes under idealistic assumptions provided insights into the geometric constraints of the line drawing graph. It was shown that the reconstruction rules derived from those geometric constraints are necessary conditions but not sufficient conditions for unique reconstruction of the LDG. The example, where unique reconstruction could not be made, points out that wing representation is not mathematically adequate for reconstructing a perfect LDG. Progress toward automatic generation of the LLDG for polygonal scenes was made by our heuristic reconstruction algorithm. The algorithm was shown by examples to be comparable to the deterministic algorithms

when the input wing set was complete. When the input wing set was not complete, a partial LLDG was generated. Difficulties in generalizing the reconstruction algorithm to include quadric scenes were identified and possible solutions were offered. Much more needs to be done to achieve the goal of automatic generation of the LLDG for quadric scenes.

One final small contribution of this thesis is the successful **fusion of registered range and intensity images** to subpixel accuracy from source images in two different world coordinate frames. The fused image provides complementary information about the surfaces in view. Object recognition related processing may benefit from the use of fused data as demonstrated by our research on wing detection and fused fitting.

7.3 Recommendations for Future Research

We have presented algorithms with examples that show line drawing generation and interpretation is plausible in the planar surface domain. Extending the implementation to handle quadric or even more general surface scenes will be an important future research topic. Some of the difficulties involved are already discussed in Chapter 6. The question "Is wing representation adequate for line drawing recovery and interpretation of quadric surfaced scenes?" needs to be addressed before reaching for a quadric surface scene reconstruction module.

The use of fused fitting enabled a more accurate recovery of wing features within a subimage of the scene; but a poor decision on the "busyness" of a subimage can result in having no wing detected on an edge contour (missing wing) or having spurious wings. Our experiments have suggested that a fixed-scale window for wing detection is inadequate across scenes with multi-size features. Variable size windows are needed to achieve a more robust wing sensor. Toward this end, a scaled-space approach to wing

detection is an important theme to be explored in the future. Scale space techniques for feature extraction have been successfully implemented in the past [35, 96, 106]. A similar feat should be duplicated to achieve a variable scale wing sensor.

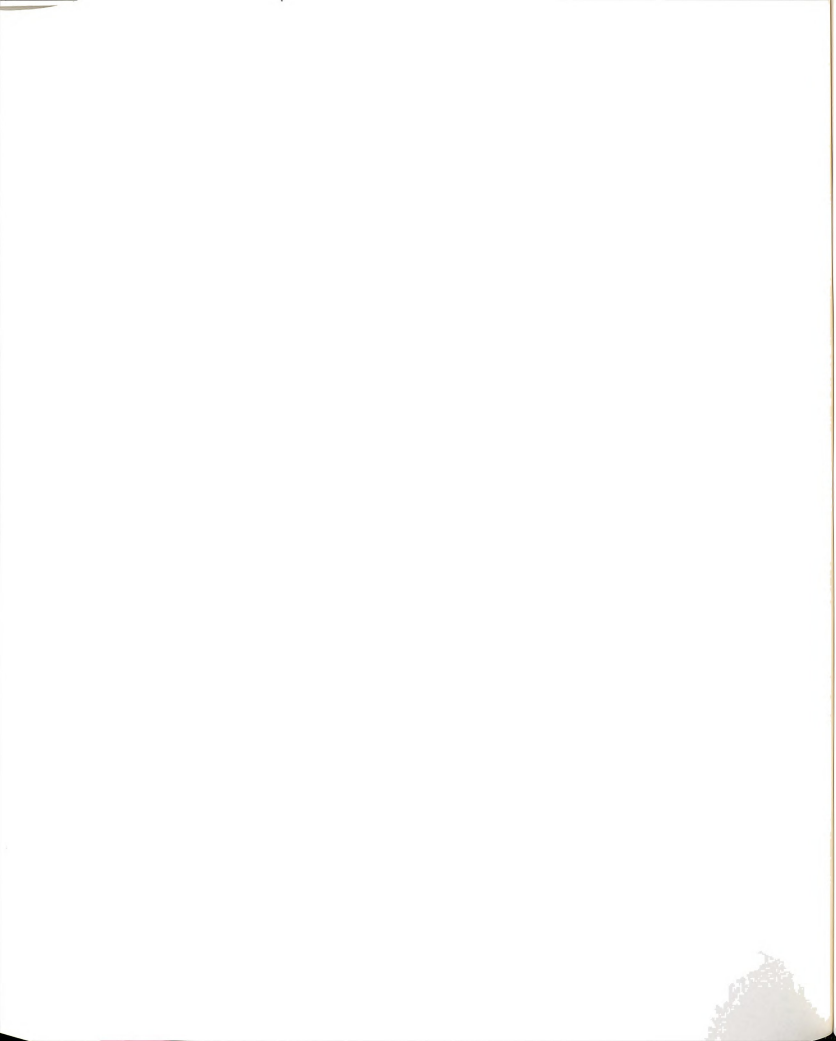
Having demonstrated the feasibility of wing detection from fused imagery, a natural improvement and extension is to forego the range image and to attempt to extract wing features using shading data alone. Intensity can be directly related to surface patch shape via the same model parameters and hence new fitting equations, such as those of Section 3.4, can be derived. Future experiments may show that rough object shape can still be distinguished by applying the same fitting method to observations derived from a single intensity image.

An immediate future goal is the parallel implementation of the wing detection and the LLDG reconstruction algorithms. Parallel implementations of our algorithms are natural. Detection of individual wings is independent of one another; once the image is tessellated into subimages, wings may be detected in parallel. With parallel wing detection, wings detected in one window with confidence may be used to aid the recovery of the wings in neighboring windows. The LLDG reconstruction procedure can also be parallelized by reconstructing each planar group of wings simultaneously. By allowing inter-group communications during reconstruction, ambiguous interpretations may be resolved during the reconstruction phase (recall the front face of the Big-Block example in Figure 4.18). The rules governing conflict resolution must be carefully constructed to ensure only a physically realizable interpretation survives.

Finally, with the advent of our wing detector, object recognition based on wing representation theory should be studied further than previously [26] done. Aspect graph representation offers a structural multi-view representation of the objects, while wing representation explicitly defines the characteristics of the visible surfaces of the object in a given view. By building an aspect-wing graph object model database, object recognition can be performed based on the set of detected wings. Furthermore,

successful partial line drawing reconstruction from wings should be most useful in object recognition tasks where the junctions in the LLDG may be used as composite wings to reduce the prohibitive large space requirement imposed by the aspect-wing graph.

APPENDICES



APPENDIX A

Creation of Fused Imagery

In this Appendix, the process of obtaining registered range and intensity imagery will be detailed. We will describe the the imaging system and its set-up follow by the complete description of the data acquisition procedure including all the necessary frame transformations.

For convenience in its use, the data is presented as if it were formed from the scene using parallel projection along the optical axis of the camera. However, a perspective transformation was needed in order to map 3D world surface points onto the intensity array produced by the camera, so that intensities could be assigned to 3D surface points. The resulting data is definitely not 3D, but rather $2\frac{1}{2}$ D, and is very much viewpoint dependent. Because the Technical Arts Scanner is a triangulating sensor, there are voids in the data due to shadowing from both the camera and laser sheet. Shadowing from the white light sources has been minimized by using two or three light sources when taking the intensity images.

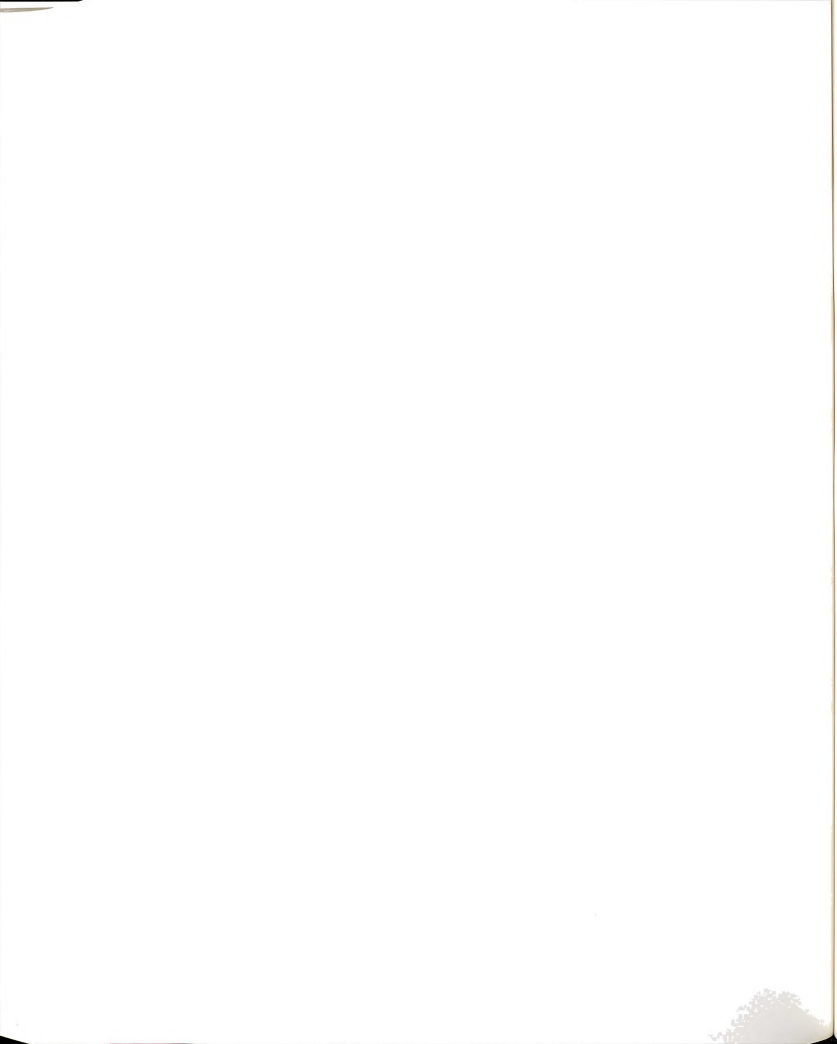
Care has been taken to assure registration accuracy to within one pixel in the final fused images. The fusion process is briefly outlined first; complete details follow.

Data Creation Process

1. Obtain range image (R_w) using Technical Arts White Scanner.
2. Obtain intensity image (I_c) using Innovision's color imaging system through White Scanner's camera.
3. Compute the calibration matrix for mapping a surface point in R_w to its corresponding point in I_c .
4. Compute the transformation matrix for transforming a point in world coordinate frame to camera coordinate frame.
5. Parallel project each surface point in R_w into the fused image plane ($z_c = 0$ plane in camera-centered frame), from which depth and intensity values can easily be computed.
6. Set the binary flag image to indicate the validity of the range/intensity values at each element in the image.

System Setup and Capturing Range Images

Figure A.1 depicts the set up of the White Scanner for capturing range images in our facility. The White Scanner is a triangulation based range scanner [117]. The stage for holding the objects in the scene can be moved in two directions: along the X_w - and Y_w -axes. A sheet of laser light is projected onto the scene and the laser stripe that hits the surface of the objects in the scene is picked up by the camera. It is those surface points (along the stripe) whose z -values (z_w) are computed using the world coordinate system. Note that only those surface points that are illuminated by the laser light and are visible to the camera can yield fused data values. The stage was moved in the direction along the X_w -axis to ensure that the z_w 's are computed for



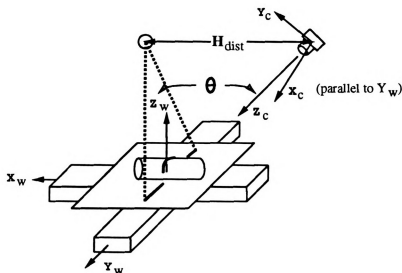
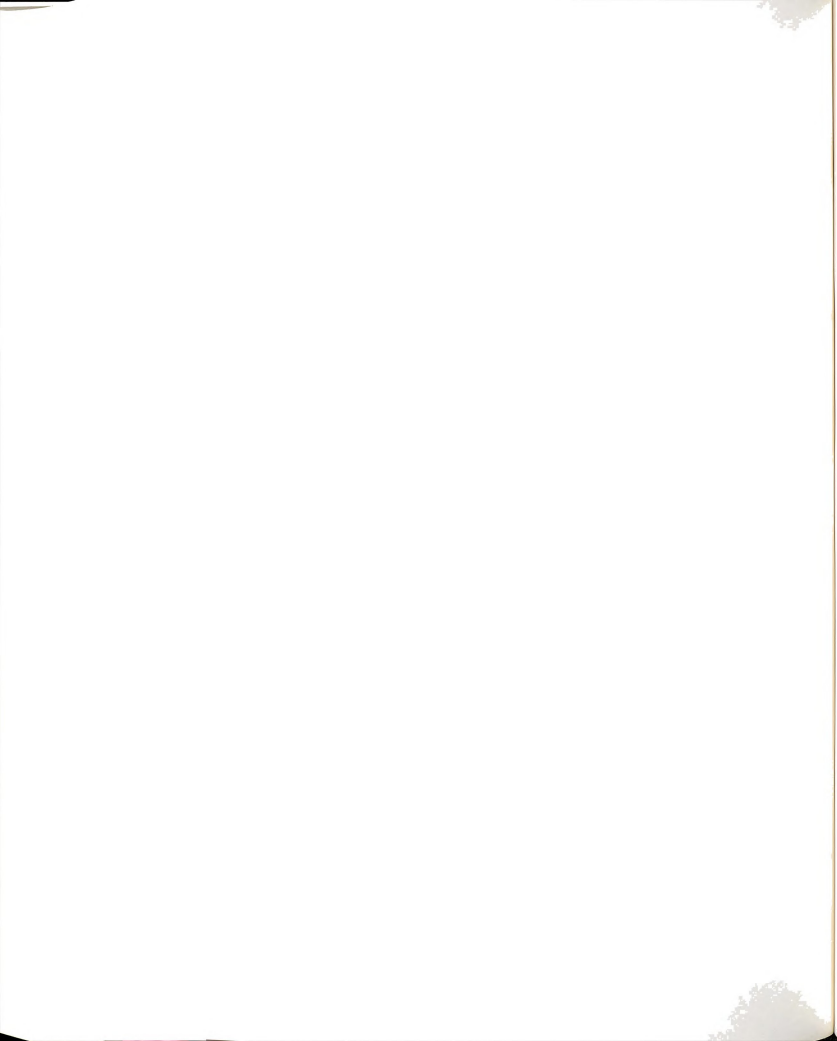


Figure A.1. Setup of White Scanner and the two coordinate frames

all *visible* surface points. The total distance moved along the X_w -axis and the strip seen by the camera were both quantized into 240 steps so that the resulting range image has the dimension of 240 pixels by 240 pixels. Since the final fused image contains both range and intensity data we use either term *pixel* or *rangal* rather than coin another term such as *fusel* ! However, the actual x_w and y_w coordinates were recorded rather than the row and column numbers. An example range image is given in Figure A.2 (a).

θ , which is the angle between the sheet of laser light and the optical axis of the camera, is fixed to be 45 degree. H_{dist} , which is the normal distance from the camera to the sheet of laser light, is known from calibrating the White Scanner beforehand. These two parameters enable the scanner to compute the z_w coordinate for each of the visible & illuminated surface points. Furthermore, knowledge of those two parameters is essential in parallel projecting the surface point into the final fused image plane as will be seen later.



Capturing Intensity Images

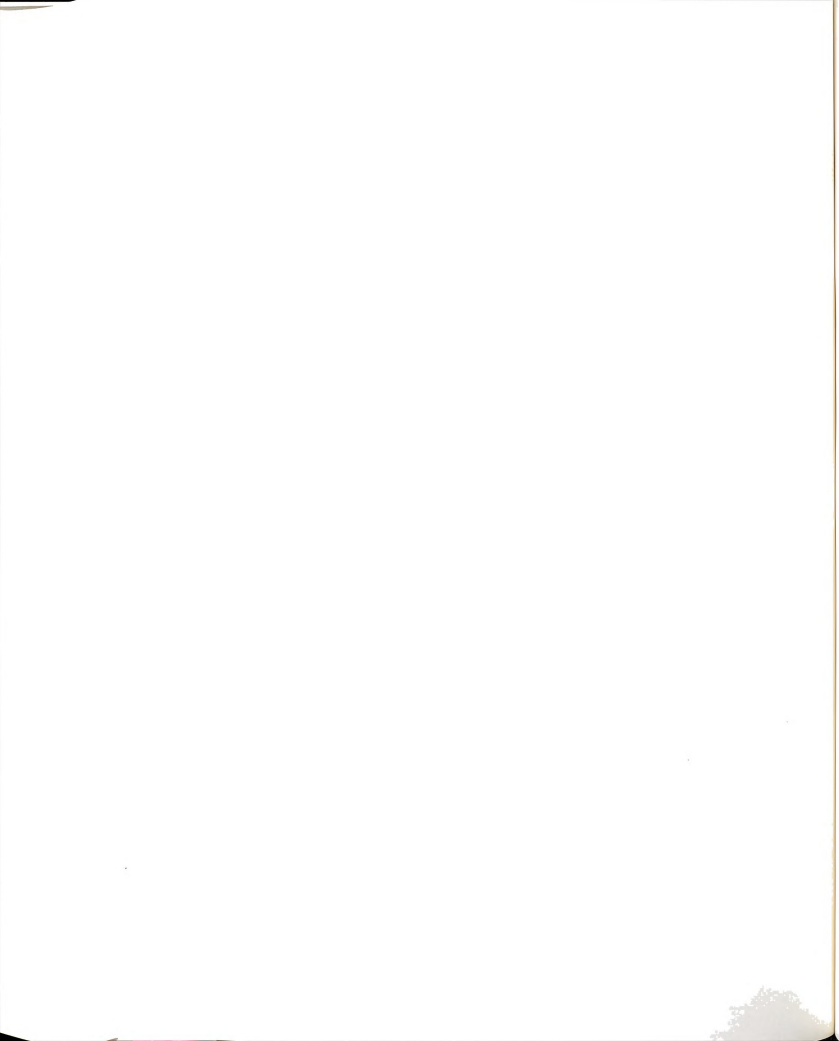
The intensity image I_c was captured through the White Scanner's camera using *digicolor*, an in house program, which drives the Datacube board for capturing images. The laser light was turned off and 1 to 3 white light sources were used to minimize the shadowed regions. The registered intensity image, originally of size 512 pixels x 485 pixels, was reduced to 240 x 240 using the "*stretch*" function (which is basically block averaging) available in the HIPS [77] package. This intensity image is camera-centered as opposed to world-centered for the range image. Note that due to occlusion of the laser stripe, some surface points that are visible in the intensity image may not have corresponding data in the range image. An example intensity image is shown in Figure A.2 (b).

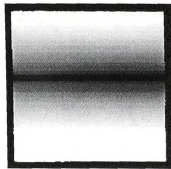
Computing the Calibration matrix

In order to obtain a fused (range and intensity) image, a calibration (transformation) matrix for mapping rangels in the range image to corresponding surface points in the the intensity image must be found. This is done by picking n , $n \geq 6$, pairs of corresponding points in the range and intensity images (say R_1, R_2, \dots, R_n and I_1, I_2, \dots, I_n) and solving for the calibration matrix C [52]. Mathematically, this process is written as

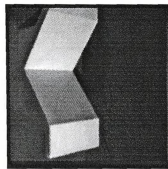
$$\begin{bmatrix} x_{R_1} & y_{R_1} & z_{R_1} & 1 \\ x_{R_2} & y_{R_2} & z_{R_2} & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_{R_n} & y_{R_n} & z_{R_n} & 1 \end{bmatrix} \cdot C_{4 \times 3} = \begin{bmatrix} x_{I_1} & y_{I_1} & 1 \\ x_{I_2} & y_{I_2} & 1 \\ \vdots & \vdots & \vdots \\ x_{I_n} & y_{I_n} & 1 \end{bmatrix}.$$

Figure A.2 (a), (b) and (c) show actual images and matching data points used in computing a calibration matrix C . Note that using this calibration matrix, subpixel accuracy was achieved for these calibration points.





(a)



(b)

Xw	Yw	Input Data				Fitted Data		Residuals	
		Zw	Xi	Yi		Xi2	Yi2	X	Y
-3.0	-1.3	-0.0	62.0	231.0		61.5	231.5	0.5	-0.5
2.9	-1.3	-0.0	133.0	215.0		133.1	214.3	-0.1	0.7
-3.0	0.6	0.0	61.0	184.0		61.5	183.8	-0.5	0.2
2.9	0.7	-0.0	133.0	172.0		133.0	172.6	0.0	-0.6
-2.9	4.2	-3.5	23.0	98.0		23.0	97.4	-0.0	0.6
2.9	4.2	-3.5	95.0	99.0		95.0	99.6	0.0	-0.6
-3.0	7.7	-0.0	61.0	11.0		61.0	11.5	-0.0	-0.5
2.9	7.7	-0.0	133.0	26.0		132.9	25.5	0.1	0.5

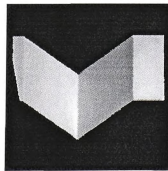
CALIBRATION MATRIX

14.92047691	-0.06266269	9.73421764	100.84538269
3.06331229	-22.63347054	-2.31218886	192.72633362
0.02696475	-0.00035766	-0.02190359	1.00000000

(c)



(d)



(e)

Figure A.2. A complete example: (a) original range image, (b) original intensity image, (c) calibration matrix, (d) fused range image, (e) fused intensity image.

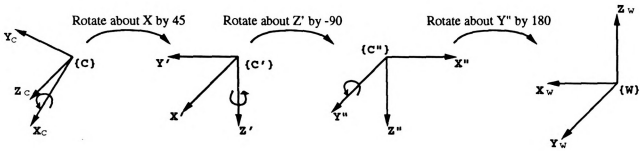


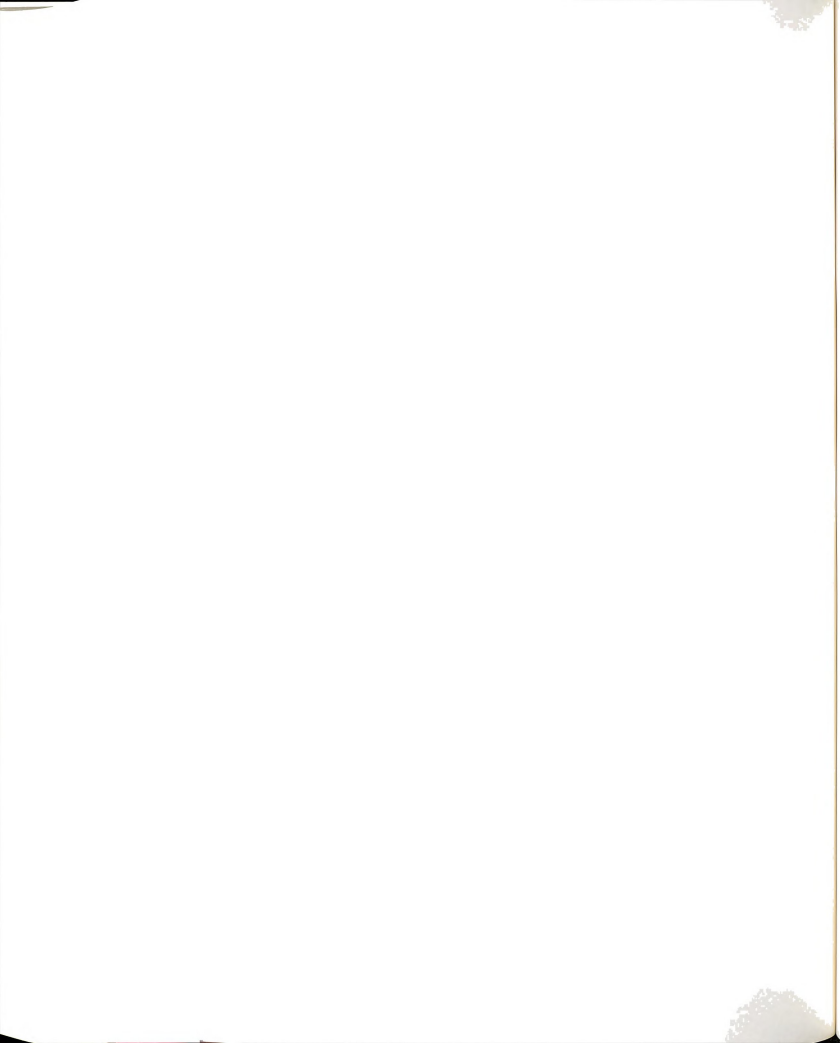
Figure A.3. Relating World coordinate frame to Camera-centered coordinate frame. To align the axes: rotate $\{C\}$ about X_c -axis by 45 degree then rotate about the new Z -axis by -90 degree follow by 180 rotation about the new Y -axis.

Forming Final Fused Images

The final step in forming the registered orthographically projected fused image is to parallel project the (x_w, y_w, z_w) into the fused image plane. The *fused image plane* is a synthetic image plane that is coplanar with the CCD array of the camera. Since the exact relationship between the world coordinate frame and the camera-centered coordinate frame is known, translate the origin of $\{W\}$ to $\{C\}$ then rotate the axis so that they align (see Figure A.3), the projection is carried out via the following transformation:

$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = R_{x_c, 45} \cdot R_{z_c', -90} \cdot R_{y_c'', 180} \cdot \begin{bmatrix} x_w - H_{dist} \\ y_w \\ z_w - H_{dist} \\ 1 \end{bmatrix}.$$

Each element in the fused image plane has 3D coordinates (x_c, y_c, z_c) , where z_c is the actual depth from image plane to object surface point. The corresponding intensity value, i_c , at (x_c, y_c, z_c) is taken from the original intensity image, I_c , at the pixel corresponding to the surface point (x_w, y_w, z_w) . This is done by projecting (x_w, y_w, z_w) into I_c using the calibration matrix derived above.

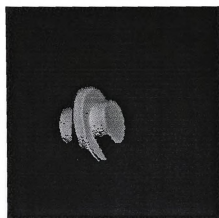


A binary flag image is used to indicate the validity of the range/intensity values at each element in the image. A pixel element is deemed to have invalid registration if at least one of the range or intensity values is not available. A background pixel will have neither range nor intensity value. A pixel has intensity but no range value when it is the projection of a surface point that is visible to the camera but not the laser. On the other hand, when computing the range values (z_w), the object(s) is(are) passed through the sheet of laser light by moving the stage along the X_w -axis direction while the intensity image is captured with the stage centered at $x_w = 0$. Thus, there may be surface points that have range values but no intensities because those points were not visible to the camera at that fixed stage position. The final fused image has 5 components (*flag*, x_c , y_c , z_c , i_c).

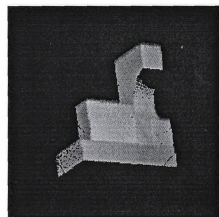
Finally, because we have forward transformed the data, there are often small holes or lines in the fused data. These are due to quantization and not shadowing. Most such holes may be easily removed using a median filter (or, by doing back transformation and interpolation). We thought that it might be better for scientific research to view the data as spatial samples which may not be dense in the array, and, thus we have not filled the holes at this time.

Fused Images

A list of 10 sample fused images are given below. These are the 10 images used to compile the wing detection summary in Table 3.7.4.



(a) agpart2

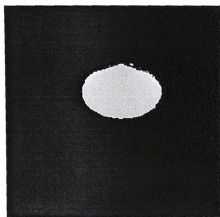


(b) block2

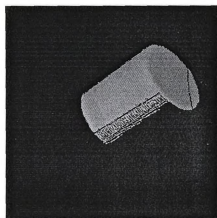
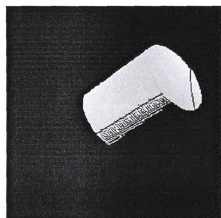


(c) bulb1

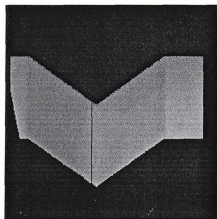
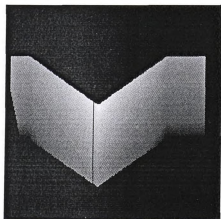
Figure A.4. cont. (see page 213 for caption)



(d) conel

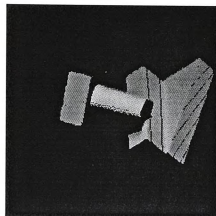
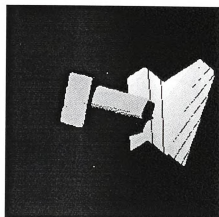


(e) cylinder2



(f) jig1

Figure A.4. cont. (see page 213 for caption)



(g) block1+column1

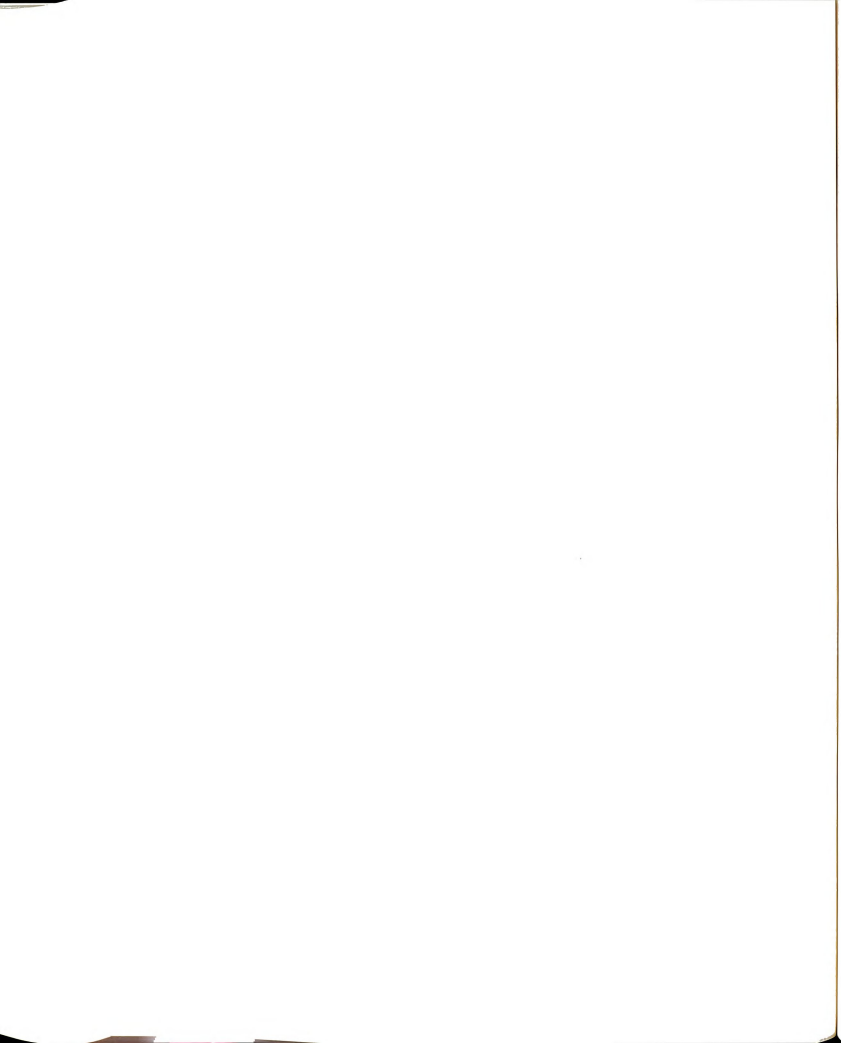


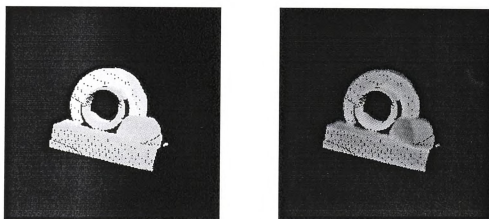
(h) cone+cylinder



(i) cone+cylinder+sphere

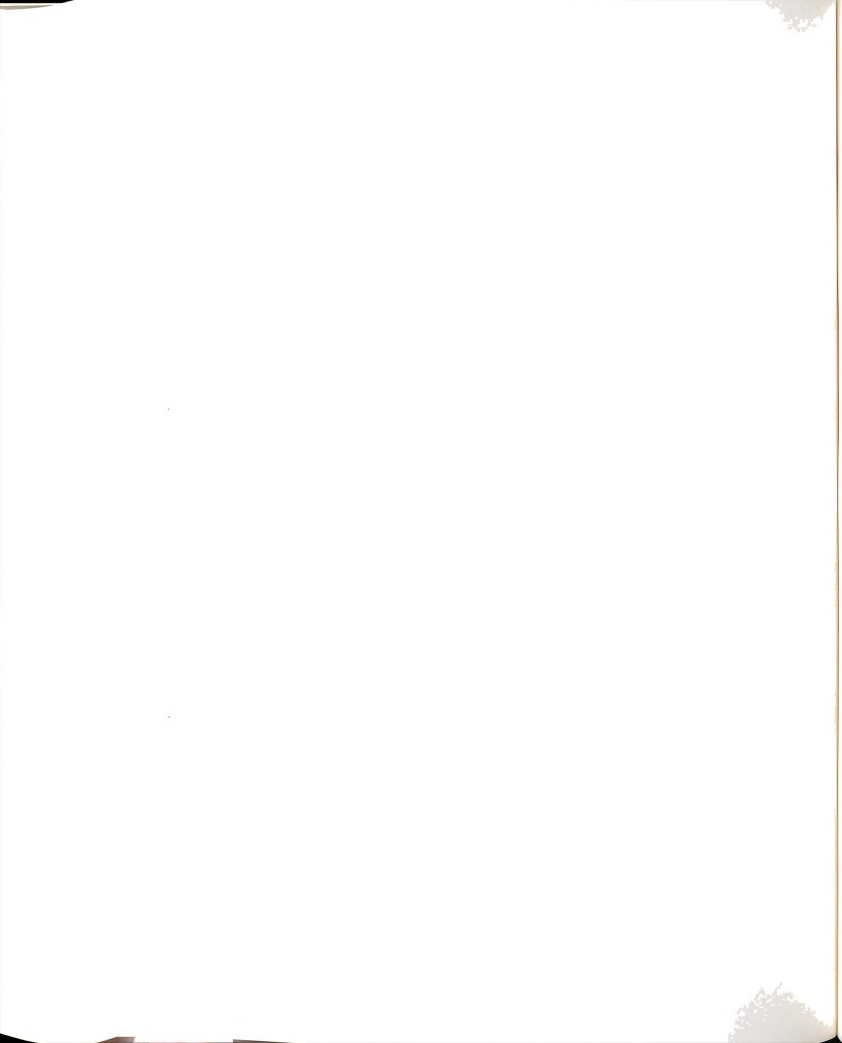
Figure A.4. cont. (see page 213 for caption)





(j) hump+agpart

Figure A.4. Fused range (left) and intensity (right) images



APPENDIX B

χ^2 Merit Function for Fused Fitting

Suppose we are fitting a set of n data points $\mathcal{X}_{s1} = \{\mathbf{x}_{s1}^1, \mathbf{x}_{s1}^2, \dots, \mathbf{x}_{s1}^n\}$ to a model $P_{3d}(\mathbf{x}; \Omega) = 0$ with m adjustable parameters Ω . Assume that for each data point $\mathbf{x}_{s1}^i \in \mathcal{X}_{s1}$, it has a measurement error that is independently and identically distributed as a normal distribution with mean 0 and standard deviation σ_{3d} :

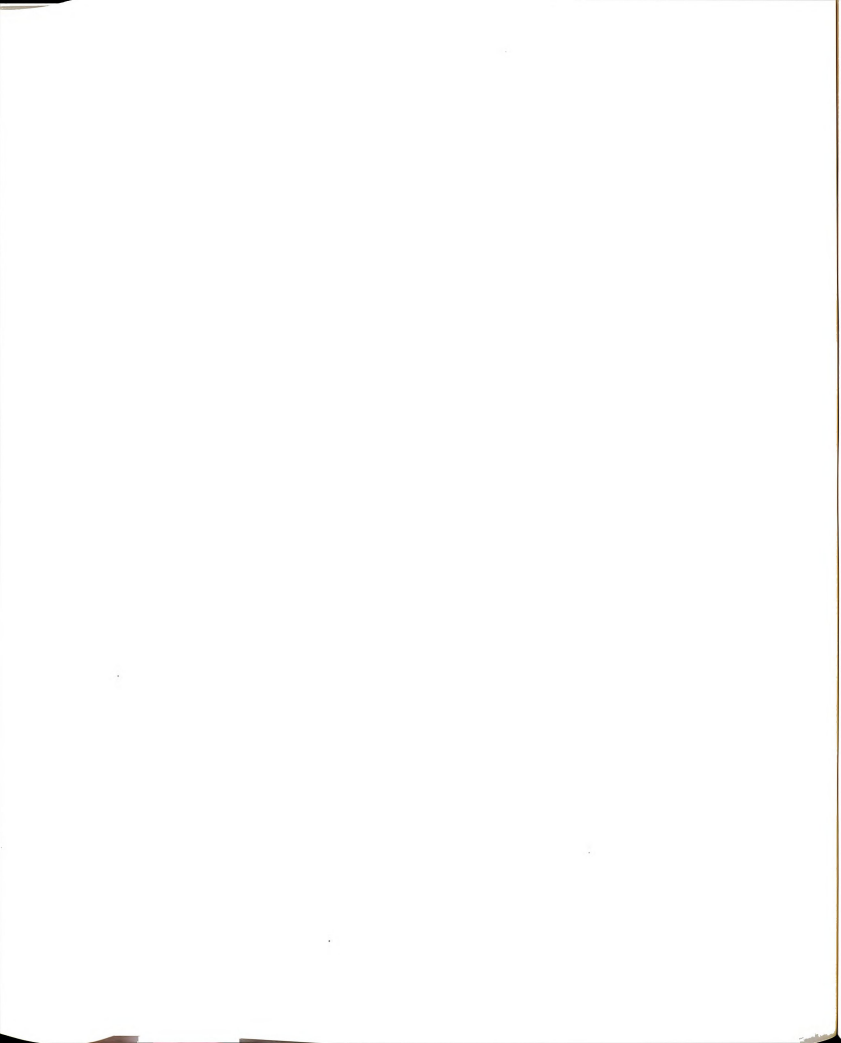
$$P_{3d}(\mathbf{x}_{s1}^i; \hat{\Omega}) \sim N(0, \sigma_{3d}^2), \quad \forall \mathbf{x}_{s1}^i \in \mathcal{X}_{s1}.$$

Then the χ^2 merit function for fitting P_{3d} is defined as

$$\chi_{n-m}^2 = \sum_{i=1}^n \left(\frac{P_{3d}(\mathbf{x}_{s1}^i; \hat{\Omega})}{\sigma_{3d}} \right)^2.$$

Given another set of n points $\mathcal{X}_c = \{\mathbf{x}_c^1, \mathbf{x}_c^2, \dots, \mathbf{x}_c^n\}$ to be fit to another model $P_{2d}(\mathbf{x}; \Omega) = 0$ with the same m adjustable parameters Ω . Again, assume that each data point $\mathbf{x}_c^i \in \mathcal{X}_c$ has a measurement error that is independently and identically distributed as a normal distribution with mean 0 but a different standard deviation σ_{2d} :

$$P_{2d}(\mathbf{x}_c^i; \hat{\Omega}) \sim N(0, \sigma_{2d}^2), \quad \forall \mathbf{x}_c^i \in \mathcal{X}_c.$$



Then the χ^2 merit function for fitting P_{2d} is defined as

$$\chi^2_{n-m} = \sum_{i=1}^n \left(\frac{P_{2d}(\mathbf{x}_c^i; \hat{\Omega})}{\sigma_{2d}} \right)^2.$$

Now let $\mathcal{X} = \mathcal{X}_{s1} \cup \mathcal{X}_c = \{\mathbf{x}_{s1}^1, \mathbf{x}_{s1}^2, \dots, \mathbf{x}_{s1}^n, \mathbf{x}_c^1, \mathbf{x}_c^2, \dots, \mathbf{x}_c^n\}$, to which we want to fit a “fused” model

$$P_{fused}(\mathbf{x}; \Omega) = a_1 P_{3d}(\mathbf{x}; \Omega) + a_2 P_{2d}(\mathbf{x}; \Omega) = 0$$

with the same m adjustable parameters, Ω , as in P_{3d} and P_{2d} , where a_1 and a_2 are two weighting constants.

Given that the fitting errors of fitting $\mathbf{x}_i \in \mathcal{X}$ to P_{3D} and P_{2D} are all *i.i.d.* normal distributed, the error of fitting weighted P_{3d} , P_{2d} and consequently P_{fused} are also normal distributed. Specifically,

$$a_1 P_{3d}(\mathbf{x}_{3d}^i; \hat{\Omega}) \sim N(0, (a_1 \sigma_{3d})^2),$$

$$a_2 P_{2d}(\mathbf{x}_c^i; \hat{\Omega}) \sim N(0, (a_2 \sigma_c)^2),$$

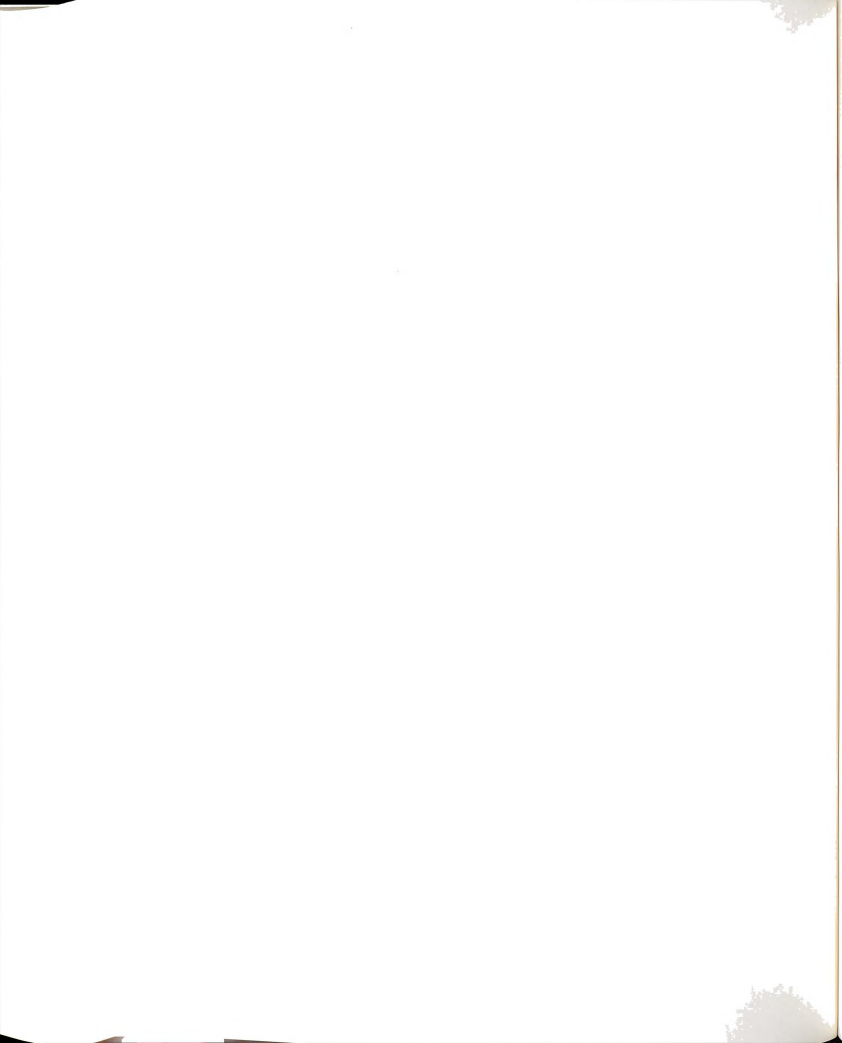
and

$$P_{fused}(\mathbf{x}^i; \hat{\Omega}) \sim N(0, (a_1 \sigma_{3d})^2 + (a_2 \sigma_c)^2), \quad \forall \mathbf{x}^i \in \mathcal{X}.$$

Therefore the χ^2 merit function for fitting a weighted fused model is

$$\chi^2_{2n-m} = \sum_{i=1}^{2n} \left(\frac{P_{fused}(\mathbf{x}^i; \hat{\Omega})}{\sigma} \right)^2.$$

where $\sigma = (a_1 \sigma_{3d})^2 + (a_2 \sigma_c)^2$.



APPENDIX C

Parameters of the Wing Detection and the LLDG Reconstruction Algorithms

Parameters of the Wing Detection Algorithm

There are two sets of parameters that are present in the wing detection algorithm: image/window size parameters and the wing detection parameters (see Table C.1). The first set of parameters control the number and the size of individual image patch for wing detection. These parameters include the image size ($N \times N$), the window size ($n \times n$), degree of overlap between adjacent windows (m rows and m columns), and the number of windows ($M \times M$). Though other image and window shapes are possible, we have selected square images and windows in this study for simplicity. Wing detection would proceed in the same way regardless of the shape of the image patch.

Two window sizes were used in the studies reported in this research: 30×30 and 20×20 pixels. The larger window size was used in compiling the statistics reported in Table 3.7.4. The wing detection results shown in Figures 3.10 and 5.7 were generated

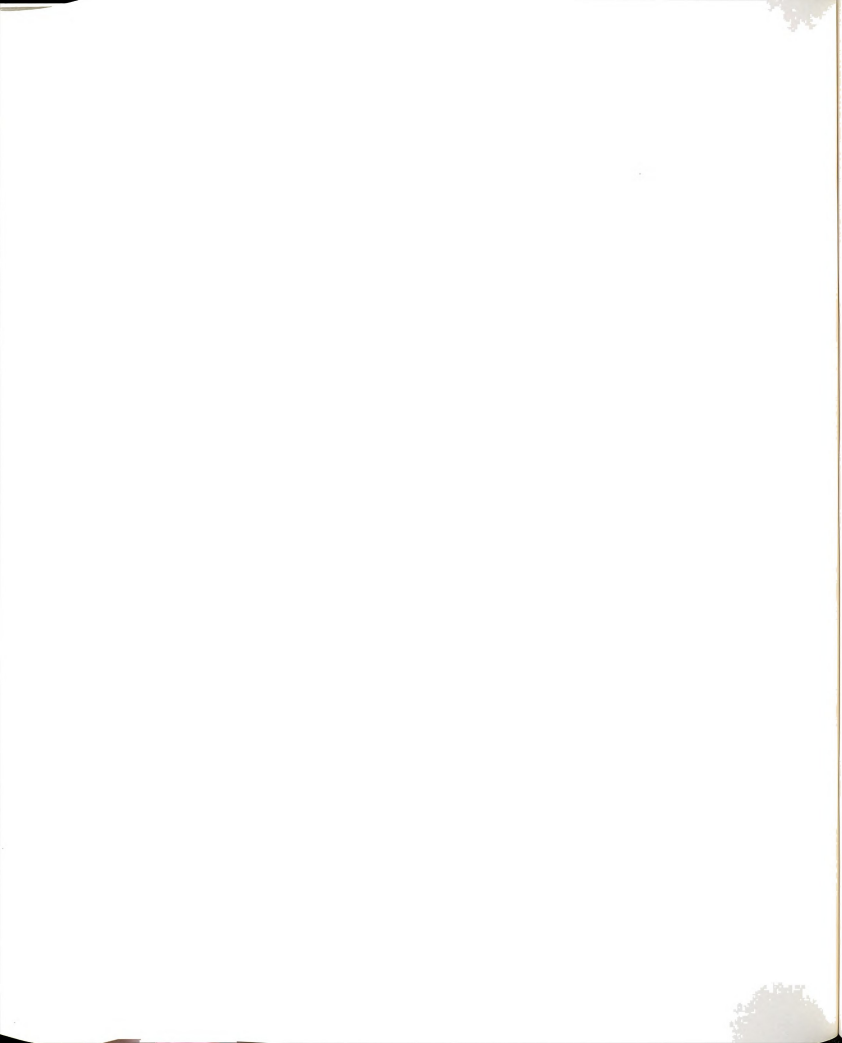


Table C.1. Parameters of the wing detection algorithm

Tesselation of the input image		
image size	$N \times N$	<i>240 × 240 pixels</i>
window size	$n \times n$	<i>30 × 30 or 20 × 20 pixels</i>
degree of window overlaps	$m \times m$	<i>15 × 15 or 10 × 10 pixels</i>
number of windows	$M \times M$	<i>225 and 529</i>
Detection of interesting windows		
initial number of potential edgelets		$2 \times n$
minimum number of edgelets	MIN-2D-PTS	n
goodness-of-fit		
linear		<i>0.1</i>
circular		<i>3.0</i>
parabolic		<i>3.0</i>
minimum number of pixels per region		$2 \times n$
Wing model fitting parameters		
total number of data points		<i>30</i>
number of 2D/3D points	n_{2D}/n_{3D}	<i>15/15</i>
number of 3D/2D/3D points	$n_{3D}/n_{2D}/n_{3D}$	<i>10/10/10</i>
number of 3D points	n_{3D}	<i>30</i>
weight distribution		
fitting 2D/3D points	w_{2D}/w_{3D}	<i>1.2/0.8</i>
fitting 3D/2D/3D points	$w_{3D}/w_{2D}/w_{3D}$	<i>0.6/0.8/0.6</i>
maximum number of iterations		<i>50</i>
χ^2 thresholds		
wing model involving only PLN surface		<i>1.5</i>
wing model involving only SPH surface		<i>2.9</i>
wing model involving only CYL surface		<i>7.5</i>
wing model involving two PLN surfaces		<i>5.0</i>
wing model involving a PLN and a CYL surfaces		<i>20.0</i>

with the smaller window. Large window size often yields fewer wing samples but more accurate parameter estimates while smaller window results in fewer missing wings but less accurate parameter estimates. The other window size related parameters were also varied as indicated in Table C.1).

The first step toward wing detection is the detection of "interesting" windows. The interesting window detecting module uses 6 parameters: the initial number of Sobel edge points used for edge contour fitting, the minimum number of Sobel edge points required on a valid edge contour, the three goodness-of-fit thresholds (experimentally chosen) for the three types of wing contours and the minimum number of valid pixels (surface points) in each of the two regions available for wing model fitting. The last parameter is to guarantee that the size of the region is adequate for sampling.

Detection of wing within each interesting window proceeds by fitting various wing models to the contour and/or surface points sampled from the window. The total number of sampled points (30) and the distribution of those points to surface and/or contour samples are some of the parameters involved in this module. If the wing model hypothesis calls for fitting of only one surface patch, then 30 data points are sampled from the surface patch. If the contour and a surface patch are to be fitted, then 15 points are sampled from each of the contour and surface patch. And, if the contour and two surface patches are to be fitted, then 10 data points are sampled from each contour and surface patches. The weights placed on the contour and surface data points during fused fitting were 1.2 and 0.8, respectively (which were not varied in the study), and the maximum number of iterations allowed for the non-linear least-squares fit is capped at 50. The experimentally chosen χ^2 goodness-of-fit thresholds round out the set of parameters in wing detection.

Table C.2. Surface Model parameters

Planar:	a_1, b_1, d_1
Spherical:	x_0, y_0, z_0, r_0
Cylindrical:	$x_0, y_0, r_0, \alpha, \beta$
Conical:	$x_0, y_0, z_0, d_0, \alpha, \beta$

Parameters of the Surface Models

The parameters of the wing surface models used in this research are listed in Table C.2. The range of these parameters for the Monte Carlo experiments performed in Section 3.5 were previously given in Table 3.1. Note that the equations of the 2D projections of the limbs and crease edges have the same set of parameters that created those limbs/edges.

Parameters of the POLY-3 Algorithm

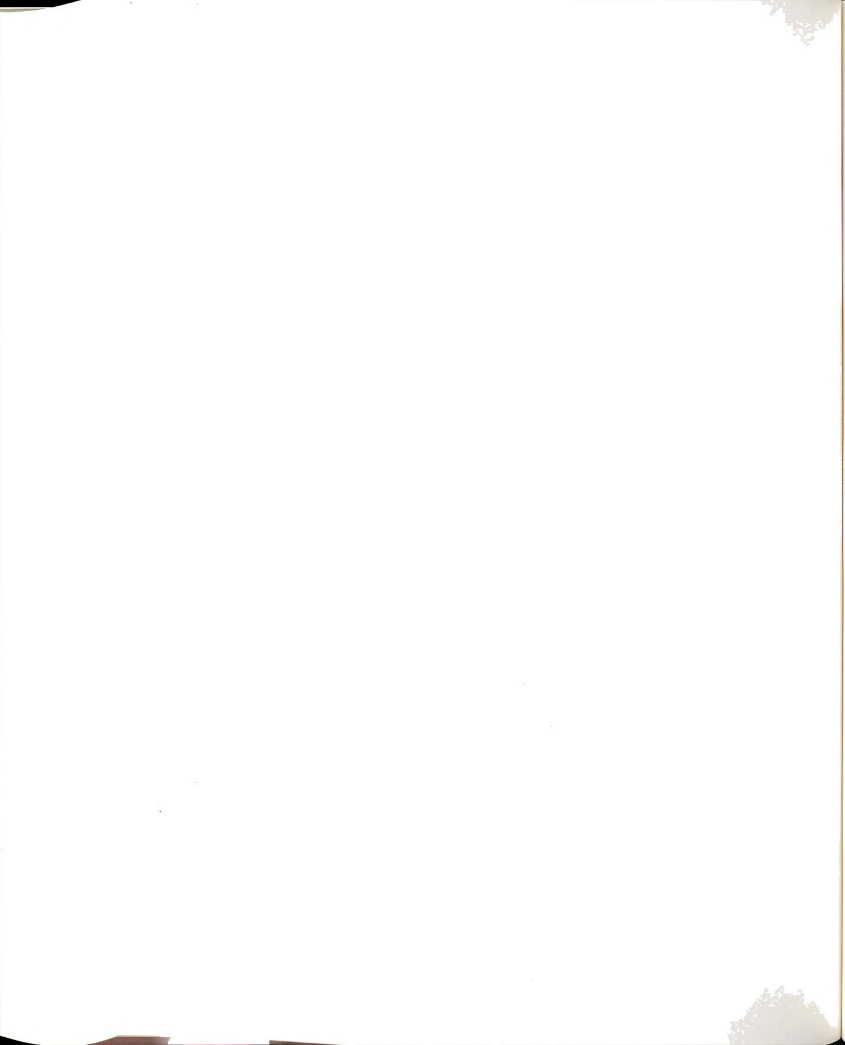
The POLY-1 and POLY-2 algorithms require no parameters due to the assumption on measurement and computation accuracy. However, in the POLY-3 algorithm, 5 parameters are needed. In preprocessing of the wing samples, short compatible wings are merged into long wings. Compatibility of two overlapping wings is based on the collinearity of the wings and the co-planarity of the adjacent surfaces. Using the polar coordinate system, 2D position of the wing can be represented by the parameters ρ and θ . Thus two wings are collinear if their respective ρ and θ values are within some thresholds. Co-planarity of surfaces are determined by requiring the surface normals and the minimum distance from the origin to the planes to fall within some predetermined thresholds. Those parameters are used again during the wing clustering step in which the planar surfaces equations of all the wings are clustered. The fifth parameter is used to cluster close-by scattered junctions in the final LDG



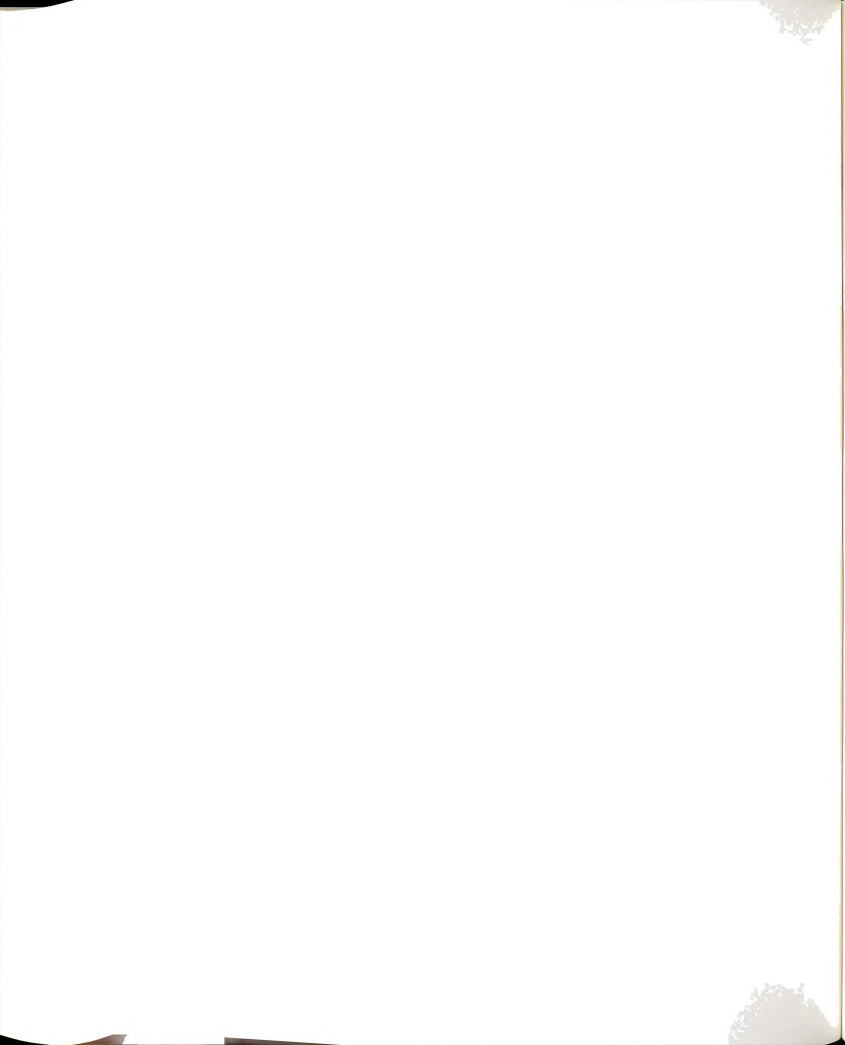
Table C.3. Parameters of the POLY-3 algorithm

collinear wing/line segments
$ \rho_1 - \rho_2 < 0.3$
$ \theta_1 - \theta_2 < 10^\circ$
coplanar surfaces
$ \text{angle between two surface normals} < 10^\circ$
$ \text{difference in distance to the origin} < 1.5$
clustering of nearby junctions
$\text{dist}(\text{jct}_1, \text{jct}_2) < 0.3$

into one representative junction after the reconstructed individual planar faces are merged. The threshold values for each of those parameters are listed in Table C.3.

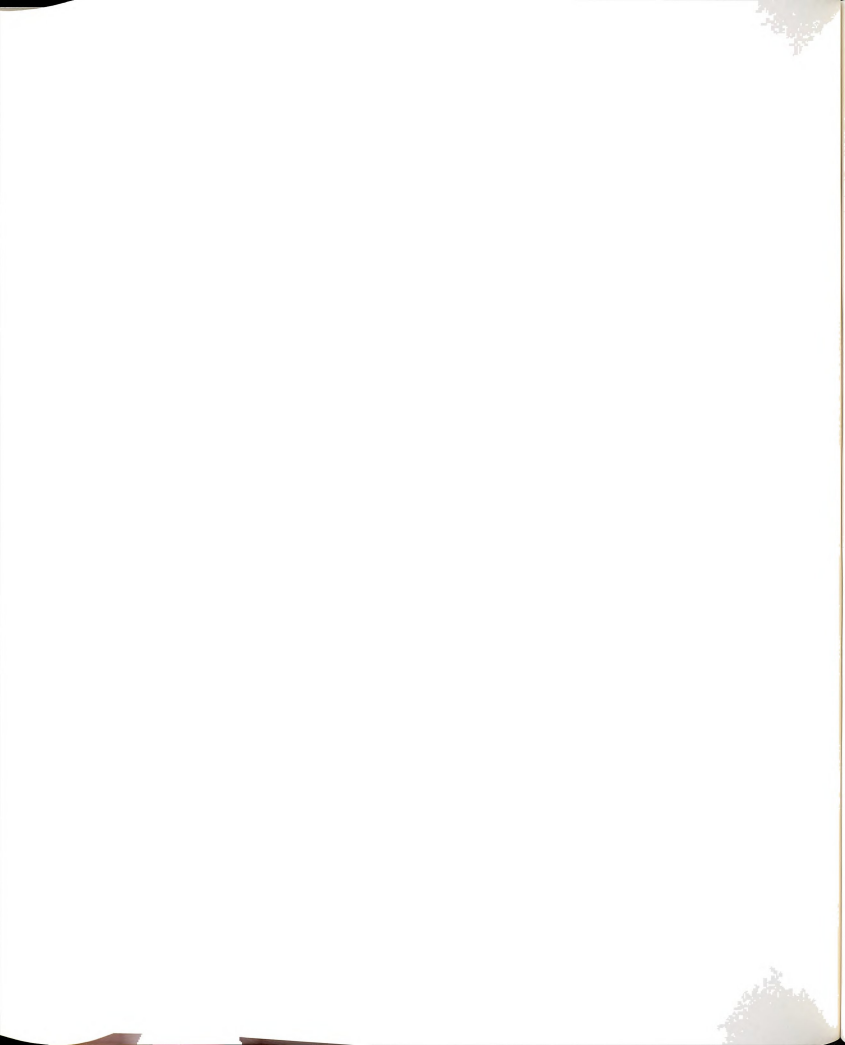


BIBLIOGRAPHY

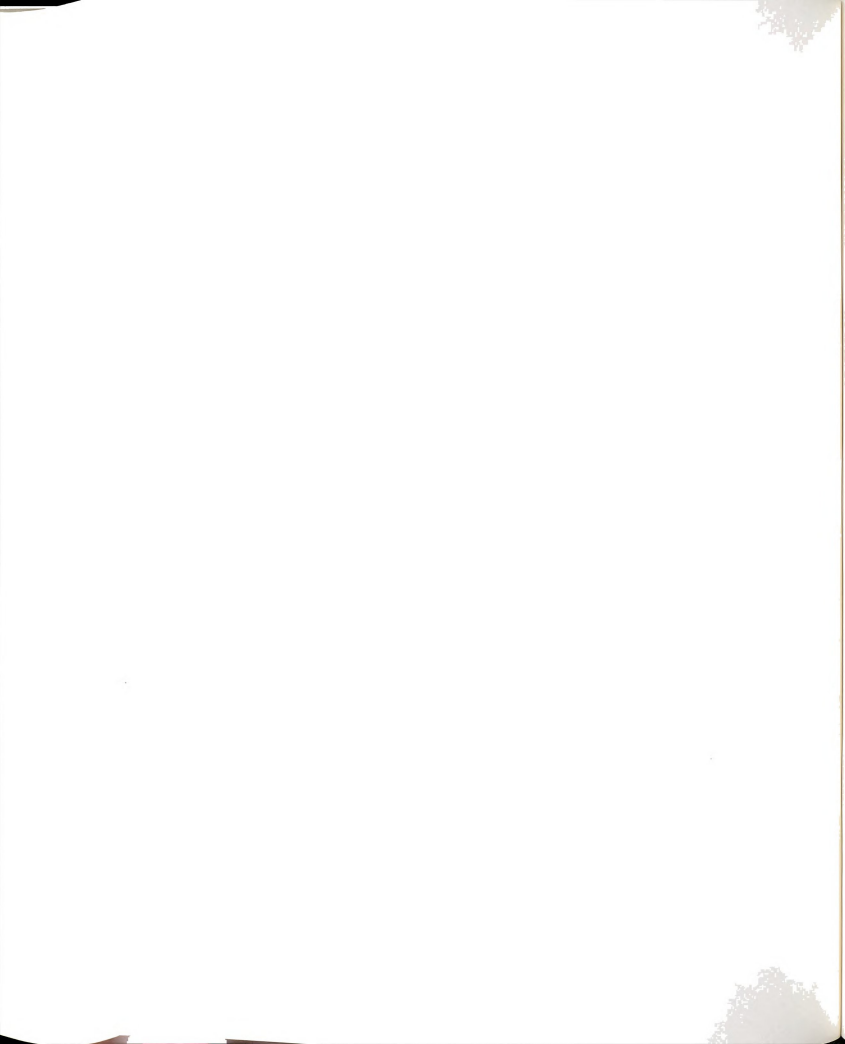


BIBLIOGRAPHY

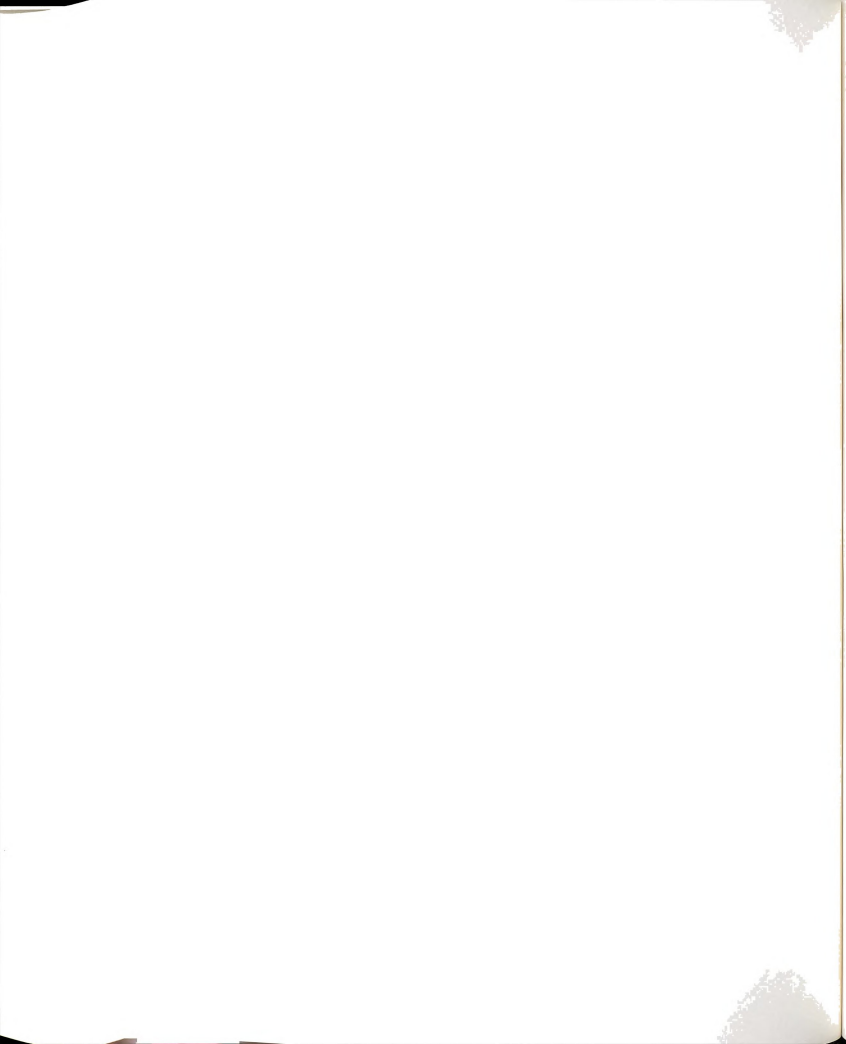
- [1] J. Aggarwal and C. Chien. 3-D structures from 2-D images. In J. L. Sanz, editor, *Advances in Machine Vision*, pages 64–121. Springer-Verlag, New York, NY, 1988.
- [2] G. Agin and T. Binford. Computer description of curved objects. In *International Joint Conference on Artificial Intelligence*, pages 629–640, October 1973.
- [3] M. Albanesi and M. Ferretti. A space saving approach to the Hough Transform. In *IEEE 1990 Conference on Computer Vision and Pattern Recognition*, pages 472–475, Atlantic City, NJ, June 1990.
- [4] I. Anderson and J. Bezdek. Curvature and tangential deflection of discrete arcs: A theory based on the commutator of scatter matrix pairs and its application to vertex detection in planar shape. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 6(1):27–40, January 1984.
- [5] R. Bajcsy and F. Solina. Three dimensional object representation revisited. In *1st International Conference on Computer Vision*, pages 231–240, London, UK, June 1987.
- [6] D. H. Ballard. Generalizing the Hough Transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122, 1981.
- [7] D. H. Ballard and C. M. Brown. *Computer Vision*. Printice-Hall, Inc., Englewood Cliffs, NJ, 1982.
- [8] S. Barnard and M. Fischler. Computational stereo. *ACM Computing Surveys*, 14(4):553–572, December 1982.
- [9] B. G. Baumgart. A polyhedron representation for computer vision. In *National Computer Conference, AFIPS*, pages 589–596, 1975.



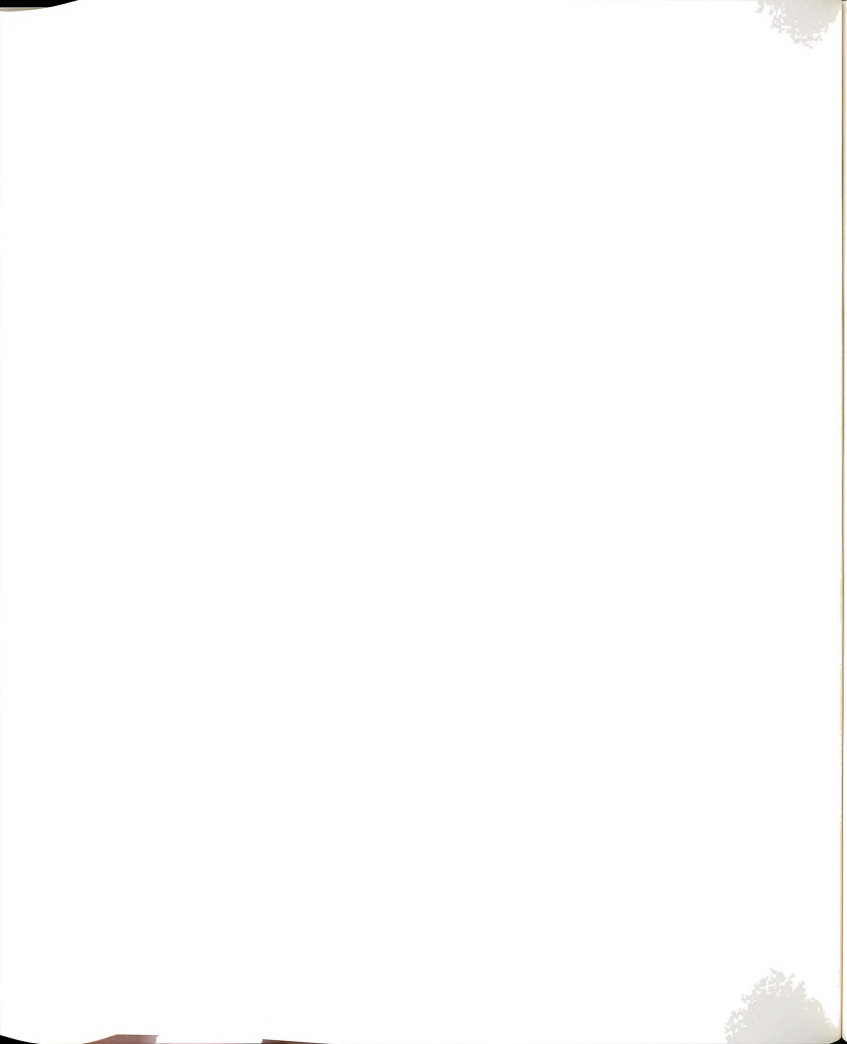
- [10] P. J. Besl. Active optical range imaging sensors. In J. L. Sanz, editor, *Advances in Machine Vision*, pages 1–63. Springer-Verlag, New York, NY, 1988.
- [11] P. J. Besl and R. C. Jain. Three-dimensional object recognition. *ACM Computing Surveys*, 17(1):75–145, March 1985.
- [12] P. J. Besl and R. C. Jain. Segmentation through variable-order surface fitting. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 10(2):167–192, March 1988.
- [13] I. Biederman. Recognition-by-components: A theory of human image understanding. *Psychological Review*, 94(2):115–147, 1987.
- [14] T. O. Binford. Visual perception by computer. In *IEEE Conference on Systems and Control*, Miami, 1971.
- [15] A. Blake and A. Zisserman. *Visual Reconstruction*. The MIT Press, Cambridge, MA, 1987.
- [16] R. Bolle and D. Cooper. On optimally combining pieces of information, with application to estimating 3-D complex-object position from range data. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 8(5):619–638, September 1986.
- [17] R. M. Bolle, A. Califano, and R. Kjeldsen. A complete and extendable approach to visual recognition. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 14(5):534–548, May 1992.
- [18] R. M. Bolle and B. Vemuri. On three-dimensional surface reconstruction methods. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 13(1):1–13, January 1991.
- [19] K. Bowyer, D. Eggert, J. Stewman, and L. Stark. Developing the aspect graph representation for use in image understanding. In *DARPA Image Understanding Workshop*, pages 831–849, Palo Alto, CA, May 1989.
- [20] K. L. Boyer and S. Sarkar. Assessing the state of the art in edge detection: 1992. In *SPIE 1708: Applications of Artificial Intelligence X: Machine Vision and Robotics*, pages 353–362, Orlando, FL, April 1992.
- [21] R. A. Brooks. Model-based three-dimensional interpretations of two dimensional images. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 5(2):140–150, March 1983.



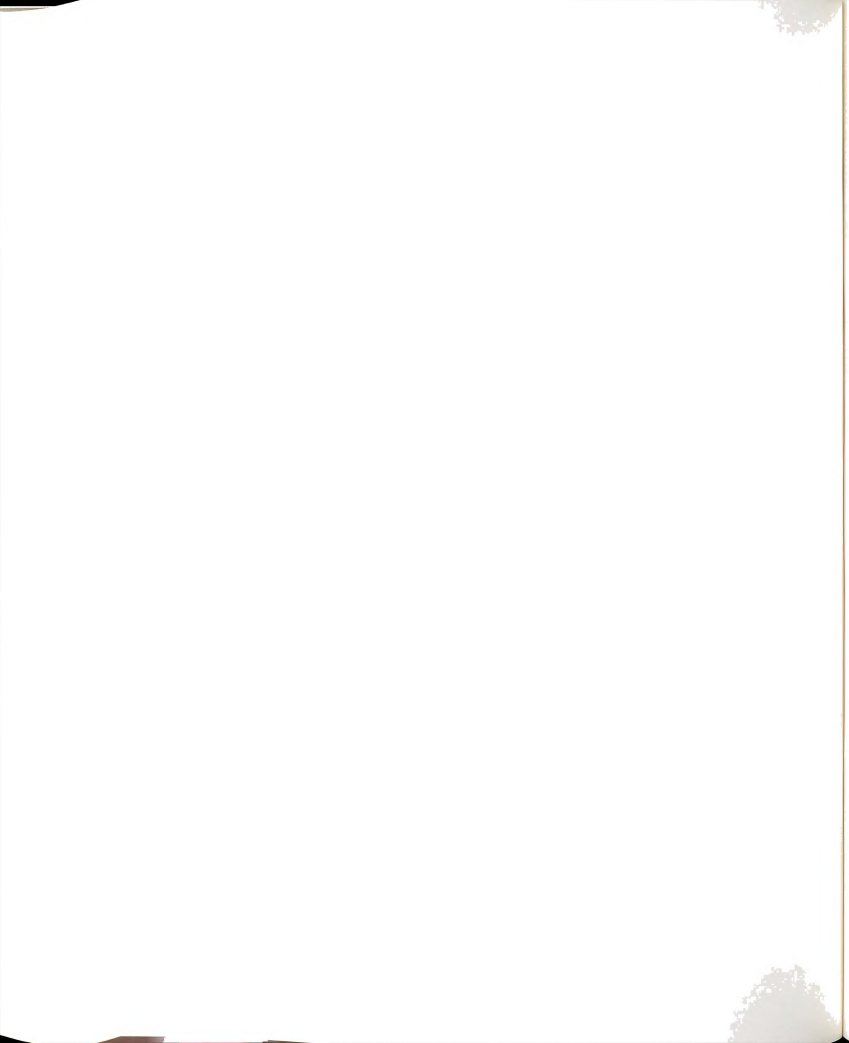
- [22] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 8(6):679-698, November 1986.
- [23] X. Cao and F. Deravi. An efficient method for multiple-circle detection. In *3rd International Conference on Computer Vision*, pages 744-747, Osaka, Japan, December 1990.
- [24] X. Cao, N. Shrikhande, and G. Hu. A comparison of quadric fitting methods for range data. Technical report, Center for Computer Vision and Robotics Research, Department of Computer Science, Central Michigan University, September 1991.
- [25] I. Chakravarty. A generalized line and junction labeling scheme with applications to scene analysis. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 1(2):202-205, April 1979.
- [26] S.-W. Chen. *3-D Representation and Recognition Using Object Wings*. Ph.D. thesis, Michigan State University, 1989.
- [27] S.-W. Chen and G. Stockman. Object wings - 2 1/2 D primitives for 3-D recognition. In *IEEE 1989 Conference on Computer Vision and Pattern Recognition*, pages 535-540, San Diego, CA, June 1989.
- [28] S.-W. Chen and G. Stockman. Wing representation for rigid 3D objects. In *10th International Conference on Pattern Recognition*, pages 398-402, June 1990.
- [29] R. T. Chin and C. R. Dyer. Model-based recognition in robot vision. *ACM Computing Surveys*, 18(1):67-108, March 1986.
- [30] M. B. Clowes. On seeing things. *Artificial Intelligence*, 2:79-116, 1971.
- [31] A. Davignon. Contribution of edges and regions to range image segmentation. In *SPIE 1708: Applications of Artificial Intelligence X: Machine Vision and Robotics*, pages 228-239, Orlando, FL, April 1992.
- [32] R. Duda and P. Hart. Use of the Hough Transformation to detect lines and curves in pictures. *Communications of the Association of Computing Machinery*, 15(1):11-15, January 1972.
- [33] R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. John Wiley, New York, NY, 1973.



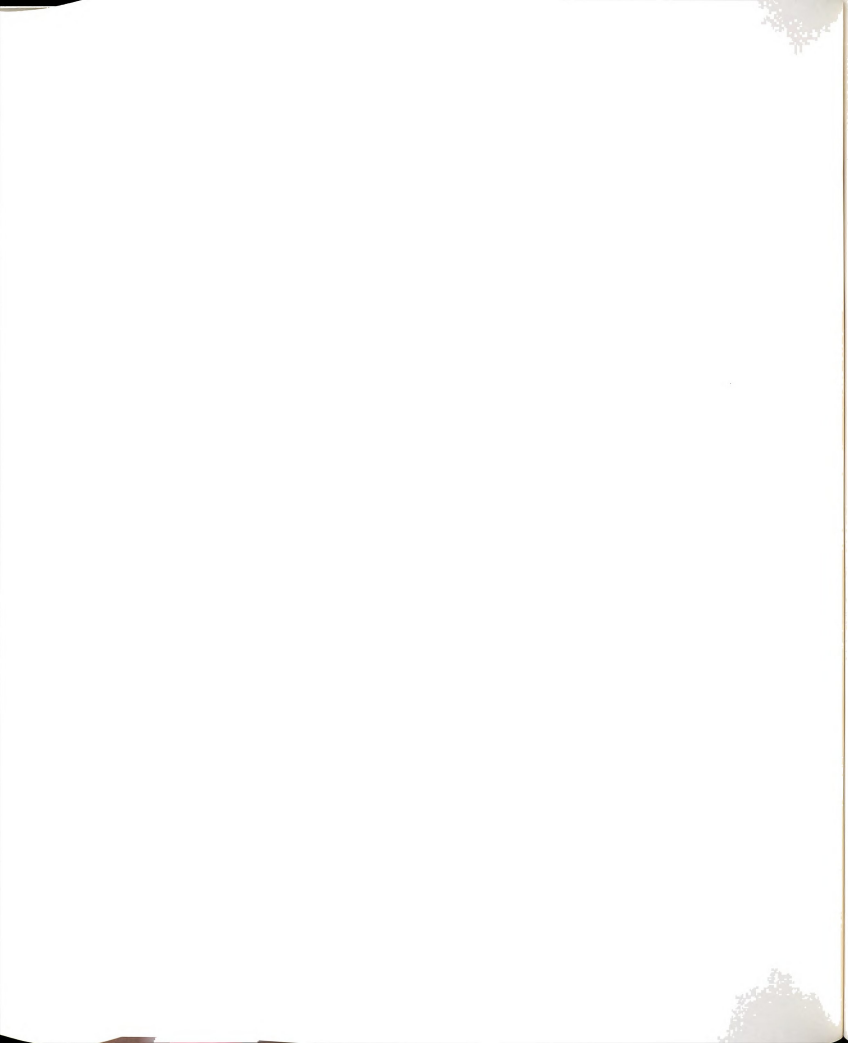
- [34] G. Falk. Interpretation of imperfect line data as a three-dimensional scene. *Artificial Intelligence*, 3:101-144, 1972.
- [35] T.-J. Fan, G. Medioni, and R. Nevatia. Segmented descriptions of 3-D surfaces. *IEEE Journal of Robotics and Automation*, RA-3(6):527-538, December 1987.
- [36] J. Fang and T. S. Huang. A corner finding algorithm for image analysis and registration. In *National Conference on Artificial Intelligence*, Pittsburgh, PA, August 1982.
- [37] O. Faugeras, M. Hebert, and E. Pauchor. Segmentation of range data into planar and quadratic patches. In *IEEE 1983 Conference on Computer Vision and Pattern Recognition*, pages 8-13, Washington, D.C., June 1983.
- [38] F. P. Ferrie, J. Lagarde, and P. Whaite. Darboux frames, snakes, and super-quadratics: Geometry from the bottom-up. In *Proceeding IEEE Workshop on Interpretation of 3D Scenes*, pages 170-176, Austin, TX, November 1989.
- [39] F. P. Ferrie, A. Lejeune, and D. Baird. Curvature, scale, and segmentation. In *SPIE 1708: Applications of Artificial Intelligence X: Machine Vision and Robotics*, pages 240-250, Orlando, FL, April 1992.
- [40] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the Association of Computing Machinery*, 24(6):381-395, June 1981.
- [41] P. J. Flynn. *CAD-Based Computer Vision: Modeling and Recognition Strategies*. Ph.D. thesis, Michigan State University, 1990.
- [42] P. J. Flynn and A. K. Jain. Surface classification: Hypothesis testing and parameter estimation. In *IEEE 1988 Conference on Computer Vision and Pattern Recognition*, pages 261-267, Ann Arbor, MI, June 1988.
- [43] P. J. Flynn and A. K. Jain. On reliable curvature estimation. In *IEEE 1989 Conference on Computer Vision and Pattern Recognition*, pages 110-116, San Diego, CA, June 1989.
- [44] H. Freeman and L. Davis. A corner-finding algorithm for chain-coded curves. *IEEE Transactions on Computers*, C-26(3):297-303, March 1977.
- [45] W. Frei and C.-C. Chen. Fast boundary detection: A generalization and a new algorithm. *IEEE Transactions on Computers*, C-26(10):988-998, October 1986.



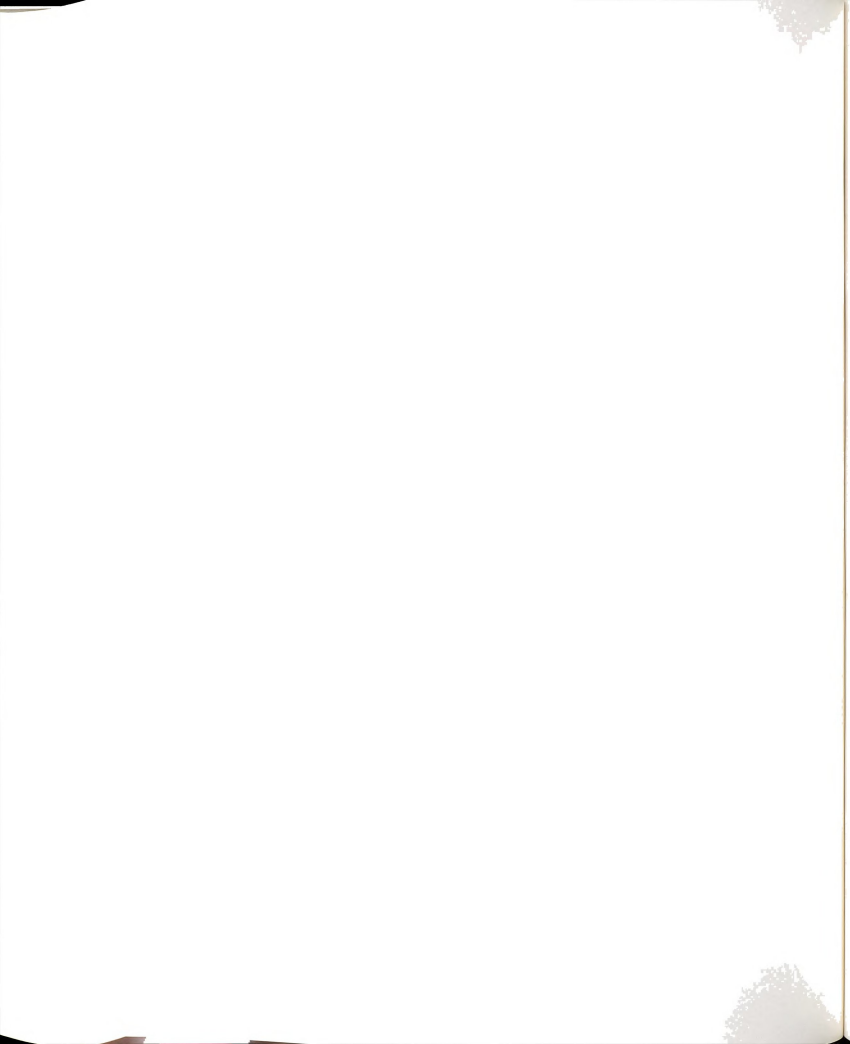
- [46] W. E. L. Grimson and D. P. Huttenlocher. On the sensitivity of the hough transform for object recognition. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 12(3):255-274, March 1990.
- [47] A. D. Gross. *Shape from a symmetric universe*. Ph.D. thesis, Columbia University, 1990.
- [48] A. D. Gross and T. E. Boulton. Error of fit measures for recovering parametric solids. In *2nd International Conference on Computer Vision*, pages 690-694, Tampa, FL, December 1988.
- [49] W. Gu and T. S. Huang. Connected line drawing extraction from a perspective view of a polyhedron. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 7(4):422-430, July 1985.
- [50] A. Gupta and R. Bajcsy. An integrated approach for surface and volumetric segmentation of range images using biquadrics and superquadrics. In *SPIE 1708: Applications of Artificial Intelligence X: Machine Vision and Robotics*, pages 210-227, Orlando, FL, April 1992.
- [51] A. Guzman. *Computer Recognition of Three-Dimensional Objects in a Visual Scene*. Ph.D. thesis, Massachusetts Institute of Technology, 1968.
- [52] E. L. Hall, J. B. K. Tio, C. A. McPherson, and F. A. Sadjadi. Measuring curved surfaces for robot vision. *IEEE Computer*, 15(12):42-54, December 1982.
- [53] D. Hoffman and W. Richards. Parts of recognition. *Cognition*, 18:65-96, 1984.
- [54] R. Hoffman and A. K. Jain. Segmentation and classification of range images. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 9(5):608-520, September 1987.
- [55] C. M. Hoffmann. *Solid and Geometric Modeling*. Morgan Kaufmann Publishers, San Mateo, CA, 1989.
- [56] B. K. Horn. Understanding image intensities. *Artificial Intelligence*, 8:201-231, 1977.
- [57] B. K. Horn. *Robot Vision*. MIT Press, Cambridge, MA, 1986.
- [58] P. Hough. Method and means for recognizing complex patterns. US Patent 3069654, 1962.



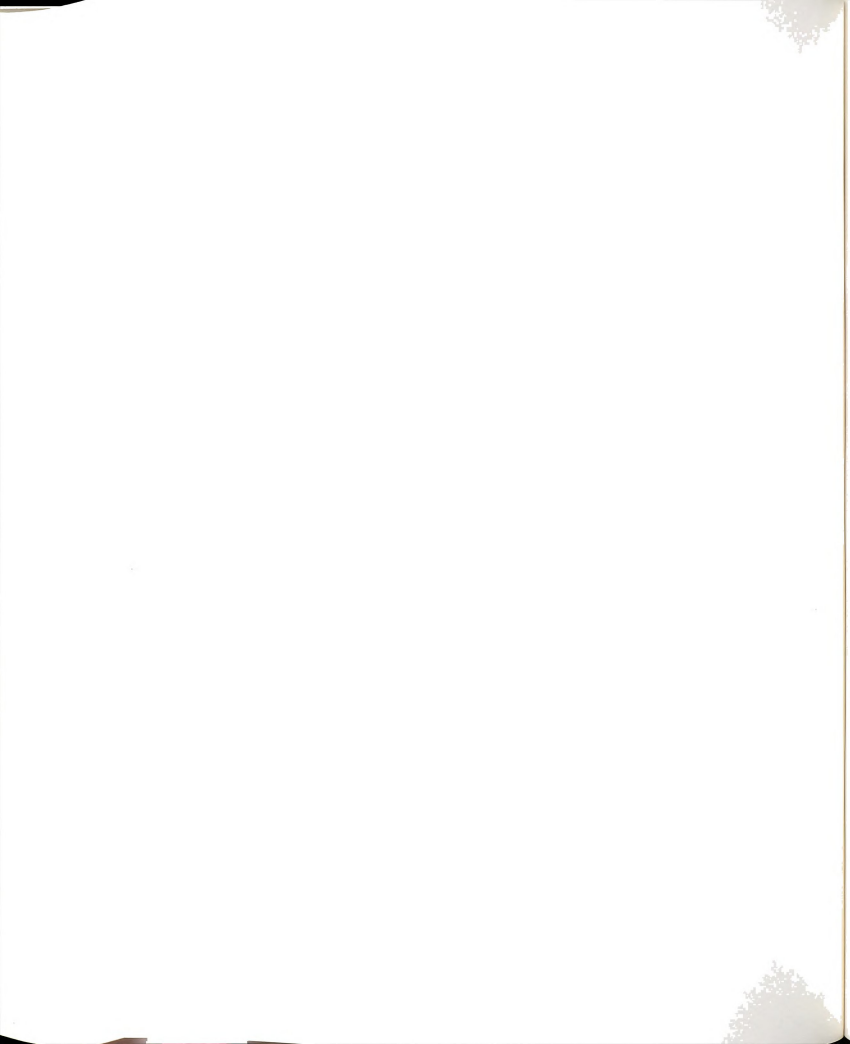
- [59] M. H. Hueckel. A local visual operator which recognizes edges and lines. *J. ACM*, 20(4):634–647, October 1973.
- [60] M. H. Hueckel. An operator which locates edges in digitized pictures. *J. ACM*, 18(1):113–125, January 1973.
- [61] D. A. Huffman. Impossible objects as nonsense sentences. In R. Meltzer and D. Michie, editors, *Machine Intelligence*, volume 8. Elsevier, New York, NY, 1971.
- [62] R. A. Hummel. Feature detection using basis functions. *Computer Vision, Graphics, and Image Processing*, 9:40–55, 1979.
- [63] D. P. Huttenlocher and S. Ullman. Recognizing solid objects by alignment. In *DARPA Image Understanding Workshop*, pages 1114–1122, Tampa, FL, Spring 1988.
- [64] J. Illingworth and J. Kittler. The adaptive Hough transform. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 9(5):690–697, May 1987.
- [65] J. Illingworth and J. Kittler. A survey of the Hough transform. *Computer Vision, Graphics, and Image Processing*, 43:221–238, 1988.
- [66] T. Kanade. A theory of origami world. *Artificial Intelligence*, 13:279–311, 1980.
- [67] T. Kanade. Recovery of the three-dimensional shape of an object from a single view. *Artificial Intelligence*, 17:409–460, 1981.
- [68] D. Keren, J. Subrahmonia, and D. B. Cooper. Robust object recognition based on implicit algebraic curves and surfaces. In *IEEE 1992 Conference on Computer Vision and Pattern Recognition*, pages 791–794, Champaign, IL, June 1992.
- [69] R. Kirsch. Computer determination of the constituent structure of biological images. *Computers and Biomedical Research*, 4(3):315–328, June 1971.
- [70] J. J. Koenderink. What does the occluding contour tell us about solid shape? *Perception*, 13:321–330, 1984.
- [71] J. J. Koenderink and A. van Doorn. Internal representation of solid shape with respect to vision. *Biological Cybernetics*, 32(4):211–216, 1979.



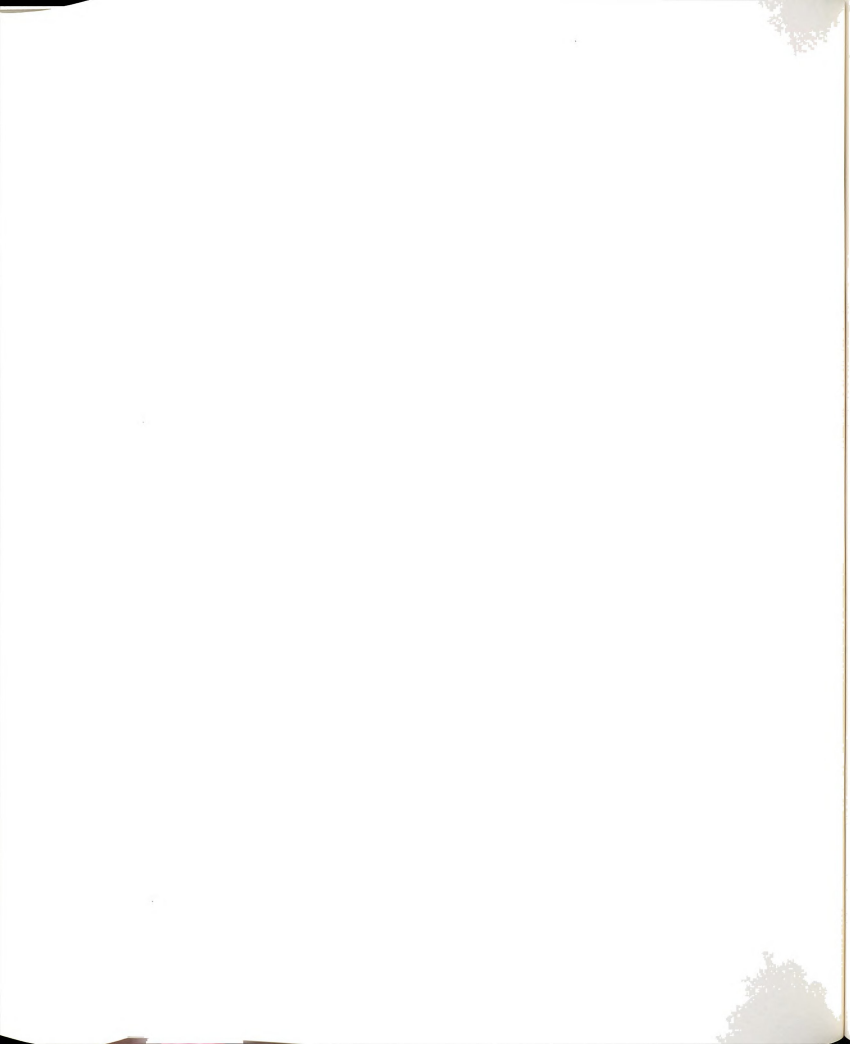
- [72] K. Koffka. *Principles of Gestalt Psychology*. Harcourt, Brace, New York, NY, 1963.
- [73] D. J. Kriegman and J. Ponce. On recognizing and positioning curved 3-D objects from image contours. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 12(12):1127–1137, December 1990.
- [74] I. Lakatos. *Proofs and Refutations*. Cambridge University Press, Cambridge, Great Britain, 1976.
- [75] Y. Lamdan and H. Wolfson. Geometric hashing: A general and efficient model-based recognition scheme. In *2nd International Conference on Computer Vision*, pages 238–249, Tampa, FL, December 1988.
- [76] P. Lancaster and K. Salkauskas. *Curve and Surface Fitting*. Academic Press, San Diego, CA, 1986.
- [77] M. Landy. *HIPS*. Human Information Processing Laboratory, NYU, New York, NY.
- [78] D. Laurendeau and D. Poussart. 3D model building using a fast range finder. In *IEEE 1986 Conference on Computer Vision and Pattern Recognition*, pages 424–426, Miami Beach, FL, 1986.
- [79] P. Liang. A new transform for curve detection. In *3rd International Conference on Computer Vision*, pages 748–751, Osaka, Japan, December 1990.
- [80] D. G. Lowe. Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, 31:355–395, 1987.
- [81] D. G. Lowe. Fitting parameterized three-dimensional models to images. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 13(5):441–450, May 1991.
- [82] A. K. Mackworth. Interpreting pictures of polyhedral scenes. *Artificial Intelligence*, 4:121–137, 1973.
- [83] J. Malik. Interpreting line drawings of curved objects. *International Journal of Computer Vision*, 1(1):73–103, 1987.
- [84] J. Malik and D. Maydan. Recovering three-dimensional shape from a single image of curved objects. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 11(6):555–566, June 1989.



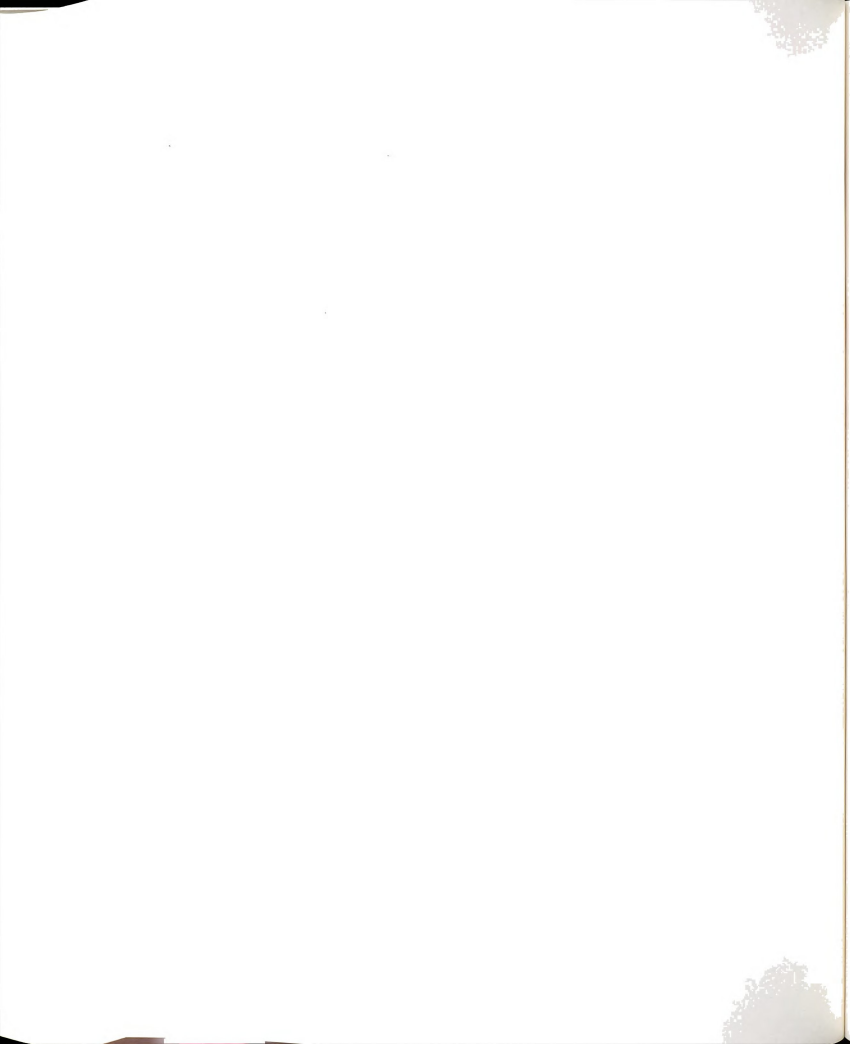
- [85] D. Marr and E. Hildreth. Theory of edge detection. *Proceedings of Royal Society of London*, 207:187–217, 1980.
- [86] D. Marr and H. Nishihara. Representation and recognition of the spatial organization of three-dimensional shapes. *Proceedings of Royal Society of London*, 200:269–294, 1977.
- [87] A. Mitiche and J. K. Aggarwal. Detection of edges using range information. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 5(2):174–178, March 1983.
- [88] H. Muammar and M. Nixon. Tristage Hough Transoform for multiple ellipse extraction. *IEE Proceedings : Part E, Computers and Digital Techniques*, 1338:27–35, 1991.
- [89] Y. Muller and R. Mohr. Planes and quadrics detection using Hough Transform. In *7th International Conference on Pattern Recognition*, pages 1101–1103, Montreal, Canada, July 1984.
- [90] S. G. Nadabar. *Markov Random Field Contextual Models in Computer Vision*. Ph.D. thesis, Michigan State University, 1992.
- [91] V. S. Nalwa. Line-drawing interpretation: A mathematical framework. *International Journal of Computer Vision*, 2(2):103–124, 1988.
- [92] V. S. Nalwa. Line-drawing interpretation: Straight lines and conic sections. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 10(4):514–529, July 1988.
- [93] V. S. Nalwa. Line-drawing interpretation: Bilateral symmetry. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 11(10):1117–1120, October 1989.
- [94] R. Nevatia and T. Binford. Description and recognition of complex-curved objects. *Artificial Intelligence*, 8:77–98, 1977.
- [95] T. S. Newman, P. J. Flynn, and A. K. Jain. Model-based surface classification. In *SPIE 1570: Geometric Methods in Computer Vision*, pages 250–261, San Diego, CA, July 1991.
- [96] B. Parvin and G. Medioni. Adaptive multiscale feature extraction from range data. *Computer Vision, Graphics, and Image Processing*, 45(3):346–356, March 1989.



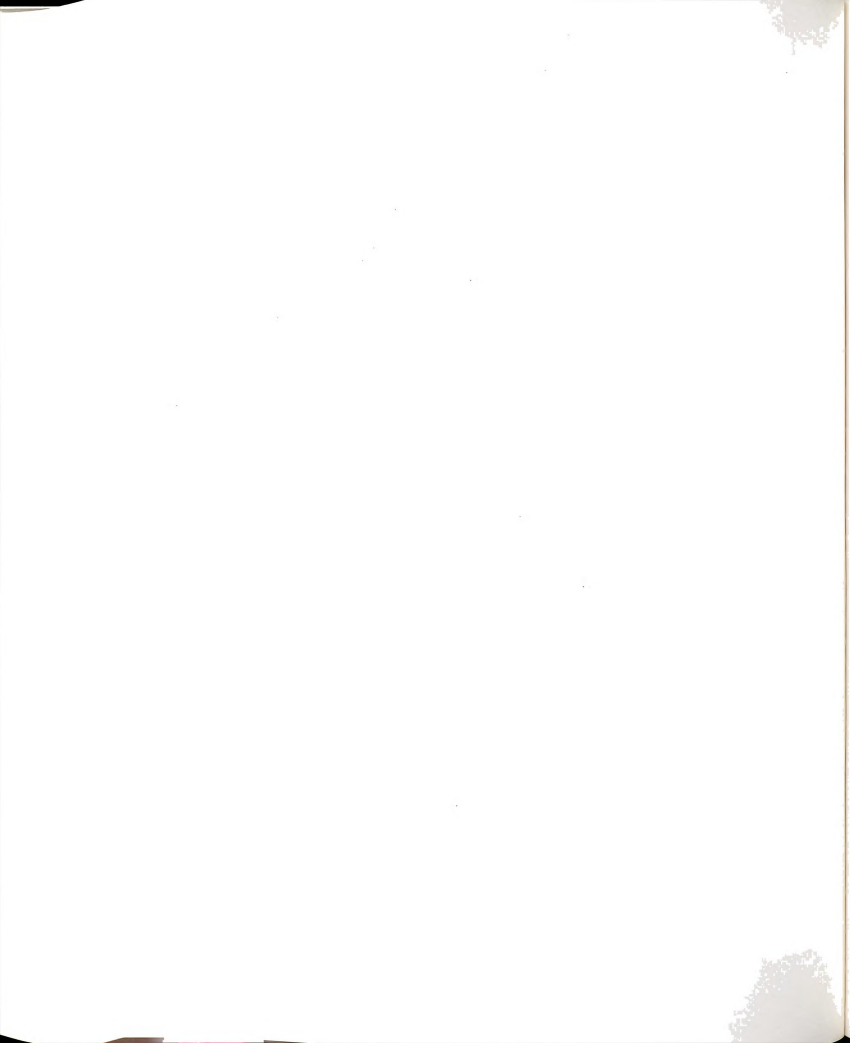
- [97] A. Pentland. Perceptual organization and the representation of natural form. *Artificial Intelligence*, 28:293–331, 1986.
- [98] A. Pentland. Recognition by parts. In *1st International Conference on Computer Vision*, pages 612–620, London, UK, June 1987.
- [99] J. Ponce and M. Brady. Toward a surface primal sketch. In *International Conference on Robotics and Automation*, pages 420–425, St. Louis, MO, March 1985.
- [100] V. Pratt. Direct least-squares fitting of algebraic surfaces. *ACM Computer Graphics SIGGRAPH'87*, 21(4):145–152, July 1987.
- [101] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling, editors. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, Cambridge, England, 1988.
- [102] J. Prewitt. Object enhancement and extraction. In B. Lipkin and A. Rosenfeld, editors, *Picture Processing and Psychopictorics*, pages 75–149. Academic Press, New York, NY, 1970.
- [103] N. S. Raja. *Obtaining Generic Parts from Range Data Using a Multi-view Representation*. Ph.D. thesis, Michigan State University, 1992.
- [104] N. S. Raja and A. K. Jain. Obtaining generic parts from range data using a multi-view representation. In *SPIE 1708: Applications of Artificial Intelligence X: Machine Vision and Robotics*, pages 602–613, Orlando, FL, April 1992.
- [105] K. Rao. *Shape Description from Sparse and Imperfect Data*. Ph.D. thesis, University of Southern California, 1980.
- [106] A. Rattarangsi and R. T. Chin. Scale-based detection of corners of planar curves. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 14(4):430–449, April 1992.
- [107] R. D. Rimey and F. S. Cohen. A maximum-likelihood approach to segmenting range data. *IEEE Journal of Robotics and Automation*, 4(3):277–286, June 1988.
- [108] L. G. Roberts. Machine perception of three-dimensional solids. In J. T. Tippett, editor, *Optical and Electro-Optical Information Processing*, pages 159–197. MIT Press, Cambridge, MA, 1965.



- [109] P. Rosin and G. West. Extracting surfaces of revolution by perceptual grouping of ellipses. In *IEEE 1991 Conference on Computer Vision and Pattern Recognition*, pages 677–678, Maui, HI, June 1991.
- [110] P. Sankar and C. Sharma. A parallel procedure for the detection of dominant points on a digital curves. *Computer Vision, Graphics, and Image Processing*, 34:321–343, 1978.
- [111] Y. Shirai. A context sensitive line finder for recognition of polyhedra. *Artificial Intelligence*, 4:95–119, 1973.
- [112] F. Solina and R. Bajcsy. Recovery of parametric models from range images: The case for superquadrics with global deformations. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 12(2):131–147, February 1990.
- [113] L. Stark. *Achieving generalized object recognition through reasoning about association of function to structure*. Ph.D. thesis, University of South Florida, Tampa, FL, 1990.
- [114] G. Stockman. Object recognition and localization via pose clustering. *Computer Vision, Graphics, and Image Processing*, 40:361–387, 1987.
- [115] K. Sugihara. *Machine Interpretation of Line Drawing*. The MIT Press, Cambridge, MA, 1986.
- [116] G. Taubin. Estimation of planar curves, surfaces, and nonplanar surface curves defined by implicit equations with applications to edge and range image segmentation. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 13(11):1115–1138, November 1991.
- [117] Technical Arts Corporation. *100X 3-Dimensional Scanner: User's Manual and Application Programming Guide*. Redmond, Washington.
- [118] C. Teh and R. Chin. On the detection of dominant points on digital curves. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 11(8):859–872, August 1989.
- [119] D. A. Trytten. *The Construction of a 2.5 Dimensional Sketch by Integration of Multiple Perceptual Processes*. Ph.D. thesis, Michigan State University, 1992.
- [120] D. A. Trytten and M. Tuceryan. Segmentation and grouping of object boundaries using energy minimization. In *IEEE 1991 Conference on Computer Vision and Pattern Recognition*, pages 730–731, Maui, HI, June 1991.



- [121] K. Turner. *Computer Perception of Curved Objects using a Television Camera*. Ph.D. thesis, School of Artificial Intelligence, University of Edinburgh, 1974.
- [122] R. Vaillant and O. Faugeras. Using occluding contours for recovering shape properties of objects. In *Proceeding IEEE Workshop on Interpretation of 3D Scenes*, pages 26–32, Austin, TX, November 1989.
- [123] D. Waltz. Understanding line drawings of scenes with shadows. In P. H. Winston, editor, *The Psychology of Computer Vision*. McGraw-Hill, New York, NY, 1975.
- [124] M. Wertheimer. Investigations on the gestalt theory. In W. Ellis, editor, *A Source Book of Gestalt Psychology*, pages 71–88. Harcourt, New York, NY, 1938.
- [125] P. H. Winston. *Introduction to Artificial Intelligence, 2nd Edition*. Addison-Wesley, New York, NY, 1984.
- [126] A. Witkin. Recovering surface shape and orientation from texture. *Artificial Intelligence*, 17:17–45, 1981.
- [127] G. Xu and S. Tsuji. Inferring surfaces from boundaries. In *1st International Conference on Computer Vision*, pages 716–720, London, UK, June 1987.
- [128] N. Yokoya and M. D. Levine. Range image segmentation based on differential geometry: A hybrid approach. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 11(6):643–649, June 1989.





MICHIGAN STATE UNIV. LIBRARIES



31293009032669