INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UMI[®]

Geometric Probing for 3D Object Recognition in Dense Range Data

Michael Greenspan

*A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for
the degree of Doctor of Philosophy*

Ottawa-Carleton Institute for Electrical and Computer Engineering
Department of Systems and Computer Engineering
1125 Colonel By Drive
Ottawa, Canada, K1S 5B6

July 1999

Copyright ©1999 by Michael A. Greenspan

National Library
of Canada

Acquisitions and
Bibliographic Services

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale
du Canada

Acquisitions et
services bibliographiques

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

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

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

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

0-612-48330-4

Canada

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

Geometric Probing
for 3D Object Recognition
in Dense Range Data

submitted by

Michael A. Greenspan, B.Sc., B.A.Sc., M.A.Sc.

in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

Professor R.A. Goubran, Chair, Department of
Systems and Computer Engineering

Professor B. Pagurek, Thesis Supervisor

Dr. P. Boulanger, Thesis Co-Supervisor

Dr. P. Besl, External Examiner

Carleton University

22 June 1999

Abstract

A new method is presented for the efficient and reliable recognition of 3D objects in dense range image data. The method is based upon a minimalistic geometric probing strategy which hypothesizes the intersection of the object with some selected image point, and searches for additional surface data at locations relative to that point. At each increment, the image is queried for the existence of surface data at a specific spatial location, and the set of possible object poses is further restricted. Eventually, either the object is identified and localized, or the initial hypothesis is refuted.

The strategy is implemented in the discrete domain as a binary decision tree classifier. The tree leaf nodes represent individual voxel templates of the model. The internal tree nodes represent the union of the templates of their descendant leaf nodes. The union of all leaf node templates is the complete template set of the model over its discrete pose space. Each internal node also references a single voxel which is the most common element of its child node templates. Traversing the tree is equivalent to efficiently matching the large set of templates at a selected image seed location.

The method was implemented and extensive experiments were conducted for a variety of combinations of tree construction and tree traversal methods under isolated, cluttered, and occluded scene conditions. The results demonstrated a tradeoff between efficiency and reliability. It was concluded that there exist combinations of tree construction and traversal which are both highly efficient and reliable.

to Mary

.

Acknowledgments

I would like to thank three groups of people.

The first is my thesis supervisors, Pierre Boulanger and Bernie Pagurek, for their guidance and many helpful suggestions.

The second group is my colleagues: Gerhard, Pierre (again), Guy, and the rest who have inspired me with their work; Bill, for introducing me to the literature on Geometric Probing; Jacques and Marc for their support.

The third group is my family members: Mary, for her unwavering support, belief, and love; Isaac, for long walks at night and various other diversions; Mom, Dad, Paula, Brian, and Lisa, for many years of respect and support. Also, Moshe, whose encouragement and example motivated me to follow this path.

Contents

List of Tables	xi
List of Figures	xiii
Glossary of Symbols	xvi
1 Introduction	1
1.1 Object Recognition, Point Probe Strategies, and Dense Range Images	1
1.2 Brief Description of Method	5
1.3 Simple Example	8
1.4 Thesis Claims	12
1.4.1 Major Claims	12
1.4.2 Minor Claims	13
1.5 Thesis Plan	14
2 Literature Review	17
2.1 Point Probe Strategies	17
2.2 Range Image Object Recognition	26
2.2.1 Correspondence Space Methods	28
2.2.1.1 Attributed Graph	28
2.2.1.2 Correspondence Tree	32
2.2.1.3 Alignment	40
2.2.2 Pose Space Methods	42
2.2.2.1 Generalized Hough Transform	42
2.2.2.2 Geometric Hashing	45
2.2.3 Recent Methods	48

2.3	Template Matching	50
2.3.1	Single Template Matching	51
2.3.2	Template Set Matching	57
2.3.3	3D Template Matching	58
2.4	Summary	60
2.4.1	Point Probe Strategies	60
2.4.2	Range Image Object Recognition	61
2.4.3	Template Matching	63
3	Description of Method	65
3.1	Recognition as a Constraint Satisfaction Problem	65
3.2	Matching by Sequential Hypothesis Testing	69
3.3	Query Point Sequence	73
3.4	Classification	74
3.4.1	Decision Tree Classification	77
3.5	Sample Tree	80
3.5.1	Structure	81
3.5.2	Decision Space	81
3.5.3	Traversal	82
3.6	Relation to Geometric Probing	83
3.7	Block Diagram	84
3.8	Summary of Chapter	86
4	Quantization	87
4.1	Decision Space Quantization	90
4.1.1	Image Voxelation	93
4.1.2	Sampling Effects	95
4.1.3	Noise Filtering	97
4.2	Pose Space Quantization	102
4.2.1	Voxel Occupancy	102
4.2.2	Viewing Sphere	104
4.3	Summary of Chapter	109
5	Class Generation	111
5.1	Stable Voxels	114
5.2	Matching Criterion	116
5.2.1	Quantization Errors	119

5.3	Matching Threshold	122
5.4	Summary of Chapter	126
6	Tree Construction	129
6.1	Background on Trees and Tree Construction	129
6.2	Optimal Methods	132
6.2.1	Dynamic Programming	133
6.2.2	Branch-and-Bound	134
6.2.3	Huffman's Algorithm	135
6.3	Sub-Optimal Methods	137
6.4	Implemented Method	138
6.4.1	Heuristic Feature Selection Criteria	138
6.4.1.1	Balance Heuristic	138
6.4.1.2	Max Information Heuristic	140
6.4.1.3	Min Information Heuristic	140
6.4.1.4	Random Heuristic	140
6.4.2	Comparison of Feature Selection Criteria	141
6.4.3	Complexity	144
6.4.4	Inclusion Lists	147
6.5	Summary of Chapter	151
7	Tree Traversal	153
7.1	Guaranteed Traversal	155
7.2	Heuristic Traversal Methods	159
7.2.1	Best-First	160
7.2.2	Beam	160
7.2.3	N-Error	161
7.2.4	Heuristic Branch-and-Bound	162
7.3	Probe Comparison	164
7.4	Summary	166
8	Experimental Results	167
8.1	Experimental Setup	167
8.1.1	Objects	167
8.1.2	Trees	168
8.1.3	Images	169
8.2	Performance Metrics	170

8.2.1	Efficiency Metrics	170
8.2.2	Reliability Metrics	170
8.3	Boat Object Tests	172
8.3.1	Correlation Threshold Determination	172
8.3.2	4 DOF Isolated Scenes	175
8.3.2.1	R_2 ANOVA	177
8.3.2.2	E_1 ANOVA	180
8.3.3	4 DOF Cluttered Scenes	183
8.3.4	6 DOF Isolated Scenes	187
8.3.5	6 DOF Cluttered Scenes	191
8.4	Duck Object Tests	194
8.4.1	4 DOF Isolated Scenes	194
8.4.2	4 DOF Cluttered Scenes	197
8.4.3	6 DOF Isolated Scenes	200
8.4.4	6 DOF Cluttered Scenes	203
8.5	Noise Robustness	206
8.6	Occlusion Robustness	208
8.7	Cascading for Enhanced Reliability	210
8.8	Parallelism for Enhanced Efficiency	212
8.9	Summary of Chapter	213
9	Conclusions	217
A	Detailed Statistical Calculations	221
A.1	Correlation Threshold Determination, $\rho = 0.015$	221
A.2	Correlation Threshold Determination, $\rho = 0.010$	223
A.3	R_2 ANOVA Calculations	223
A.3.1	ANOVA	223
A.3.2	Fisher's LSD	224
A.4	E_1 ANOVA Calculations	226
A.4.1	ANOVA	226
A.4.2	Fisher's LSD	228
A.5	Rejection Region Boundaries, $\rho = 0.015$	229
A.5.1	Positive Trial	229
A.5.2	Negative Trial	230
A.6	Rejection Region Boundaries, $\rho = 0.010$	231
A.6.1	Positive Trial	231

A.6.2	Negative Trial	231
B	Tabulated Test Results	232
B.1	Boat/4 DOF/Isolated	233
B.2	Boat/4 DOF/Cluttered	234
B.3	Boat/6 DOF/Isolated	235
B.4	Boat/6 DOF/Cluttered	236
B.5	Duck/4 DOF/Isolated	237
B.6	Duck/4 DOF/Cluttered	238
B.7	Duck/6 DOF/Isolated	239
B.8	Duck/6 DOF/Cluttered	240
C	Selected Positive Results	241
C.1	Boat, 4 DOF, Isolated	242
C.2	Boat, 4 DOF, Cluttered	244
C.3	Boat, 4 DOF, Occluded	246
C.4	Boat, 6 DOF, Isolated	247
C.5	Boat, 6 DOF, Cluttered	249
C.6	Boat, 6 DOF, Occluded	251
C.7	Duck, 4 DOF, Isolated	252
C.8	Duck, 4 DOF, Cluttered	254
C.9	Duck, 4 DOF, Occluded	256
C.10	Duck, 6 DOF, Isolated	258
C.11	Duck, 6 DOF, Cluttered	260
C.12	Duck, 6 DOF, Occluded	262
D	Paper Reprints	263
	Bibliography	290

List of Tables

4.1	Icosahedra Pose Space Sizes	109
5.1	Discrete Pose Space Sizes (n_w)	113
6.1	Duck Tree Metrics	142
6.2	Boat Tree Metrics	142
6.3	Tree Construction Complexity	148
6.4	Tree Generation Time (seconds)	150
7.1	Tree Traversal Complexity Expressions	166
8.1	R_2 Statistics ($\rho = 0.015\ m$)	173
8.2	R_2 Statistics ($\rho = 0.010m$)	175
8.3	$\bar{R}_2$ Data Summary	177
8.4	$\bar{E}_1$ Data Summary	177
8.5	R_2 LSD Partition	180
8.6	E_1 LSD Partition	181
8.7	Performance Without Cascading	211
8.8	Performance With Cascading	212
A.1	R_2 ANOVA Data	225
A.2	E_1 ANOVA Data	227
B.1	$\bar{E}_1$ Results (boat/4dof/isolated)	233
B.2	$\bar{R}_1$ Results (boat/4dof/isolated)	233
B.3	$\bar{R}_2$ Results (boat/4dof/isolated)	233
B.4	$\bar{E}_1$ Results (boat/4dof/cluttered)	234
B.5	$\bar{R}_1$ Results (boat/4dof/cluttered)	234
B.6	$\bar{R}_2$ Results (boat/4dof/cluttered)	234

B.7	$\bar{E}_1$ Results (boat/6dof/isolated)	235
B.8	$\bar{R}_1$ Results (boat/6dof/isolated)	235
B.9	$\bar{R}_2$ Results (boat/6dof/isolated)	235
B.10	$\bar{E}_1$ Results (boat/6dof/cluttered)	236
B.11	$\bar{R}_1$ Results (boat/6dof/cluttered)	236
B.12	$\bar{R}_2$ Results (boat/6dof/cluttered)	236
B.13	$\bar{E}_1$ Results (duck/4dof/isolated)	237
B.14	$\bar{R}_1$ Results (duck/4dof/isolated)	237
B.15	$\bar{R}_2$ Results (duck/4dof/isolated)	237
B.16	$\bar{E}_1$ Results (duck/4dof/cluttered)	238
B.17	$\bar{R}_1$ Results (duck/4dof/cluttered)	238
B.18	$\bar{R}_2$ Results (duck/4dof/cluttered)	238
B.19	$\bar{E}_1$ Results (duck/6dof/isolated)	239
B.20	$\bar{R}_1$ Results (duck/6dof/isolated)	239
B.21	$\bar{R}_2$ Results (duck/6dof/isolated)	239
B.22	$\bar{E}_1$ Results (duck/6dof/cluttered)	240
B.23	$\bar{R}_1$ Results (duck/6dof/cluttered)	240
B.24	$\bar{R}_2$ Results (duck/6dof/cluttered)	240

List of Figures

1.1	2D Binary Image	9
1.2	Model Templates	9
1.3	Likelihood of $p \in \mathbf{M}$ In Seed Neighborhood	10
1.4	Remaining Interpretations	11
3.1	Types of Occlusion	66
3.2	Constraints	68
3.3	Probe Order	73
3.4	Decision Space	76
3.5	Decision Tree	79
3.6	Process Flow Block Diagram	85
4.1	Quantization Function	92
4.2	Image Voxelation	96
4.3	Rank Filter	100
4.4	Plot of filter threshold (k) vs. delta voxels removed ($\Delta\ g_k(\mathbf{V})\ $)	101
4.5	Geometric Construction Relating Model Radius ($r_{\mathbf{M}}$), Voxel Space Resolution (ρ), and Angle Resolution (ε)	104
4.6	Spherical Coordinate Frame	106
4.7	Icosahedron and Derived Subdivisions	108
5.1	Stable Voxels at Various Thresholds	117
5.2	Ratio of Stable Voxels to Total Voxels (n_s/n_v) vs. Threshold (t_v)	118
5.3	Translational Pose Perturbation vs. Correlation ($t_v = 0.75$) . .	121
5.4	Rotational Pose Perturbation vs. Correlation ($t_v = 0.75$) . . .	122
5.5	Rotational Pose Perturbation vs. Correlation ($t_v = 0.50$) . . .	123
5.6	Rotational Pose Perturbation vs. Correlation ($t_v = 0.25$) . . .	124

5.7	Regression of Rotational Pose Perturbation vs. Correlation . .	124
5.8	Comparison of Pose Perturbation vs. Correlation	126
5.9	Confidence Intervals	127
6.1	Tree Construction Pseudocode	139
6.2	# of Processors (m) vs. Inclusion List Generation Time (t) . .	149
7.1	Traversal Correlation Threshold (r_t) vs. Efficiency (E_1), Guaranteed	158
7.2	Traversal Correlation Threshold (r_t) vs. Efficiency (E_1), Guaranteed and Heuristic Branch-and-Bound	163
7.3	Comparison of Probes for Different Traversals	165
8.1	Boat, 4 dof, Isolated Scene	178
8.2	Efficiency ($\bar{E}_1$) vs. Reliability ($\bar{R}_2$), (boat/4dof/isolated) . .	183
8.3	Efficiency ($\bar{E}_1$) vs. Reliability ($\bar{R}_1, \bar{R}_2$), (boat/4dof/isolated) .	184
8.4	Boat, 4 dof, Cluttered Scene	185
8.5	Efficiency ($\bar{E}_1$) vs. Reliability ($\bar{R}_1, \bar{R}_2$), (boat/4dof/cluttered) .	186
8.6	Efficiency ($\bar{E}_1$) vs. Reliability ($\bar{R}_1, \bar{R}_2$), (boat/4dof/cluttered) .	187
8.7	Boat, 6 dof, Isolated Scene	188
8.8	Efficiency ($\bar{E}_1$) vs. Reliability ($\bar{R}_1, \bar{R}_2$), (boat/6dof/isolated) .	189
8.9	Efficiency ($\bar{E}_1$) vs. Reliability ($\bar{R}_1, \bar{R}_2$), (boat/6dof/isolated) .	190
8.10	Boat, 6 dof, Cluttered Scene	192
8.11	Efficiency ($\bar{E}_1$) vs. Reliability ($\bar{R}_1, \bar{R}_2$), (boat/6dof/cluttered) .	193
8.12	Efficiency ($\bar{E}_1$) vs. Reliability ($\bar{R}_1, \bar{R}_2$), (boat/6dof/cluttered) .	193
8.13	Duck, 4 dof, Isolated Scene	195
8.14	Efficiency ($\bar{E}_1$) vs. Reliability ($\bar{R}_1, \bar{R}_2$), (duck/4 DOF/isolated) .	196
8.15	Efficiency ($\bar{E}_1$) vs. Reliability ($\bar{R}_1, \bar{R}_2$), (duck/4 DOF/isolated) .	196
8.16	Duck, 4 dof, Cluttered Scene	198
8.17	Efficiency ($\bar{E}_1$) vs. Reliability ($\bar{R}_1, \bar{R}_2$), (duck/4 DOF/cluttered) .	199
8.18	Efficiency ($\bar{E}_1$) vs. Reliability ($\bar{R}_1, \bar{R}_2$), (duck/4 DOF/cluttered) .	199
8.19	Duck, 6 dof, Isolated Scene	201
8.20	Efficiency ($\bar{E}_1$) vs. Reliability ($\bar{R}_1, \bar{R}_2$), (duck/6 DOF/isolated) .	202
8.21	Efficiency ($\bar{E}_1$) vs. Reliability ($\bar{R}_1, \bar{R}_2$), (duck/6 DOF/isolated) .	202
8.22	Duck, 6 dof, Cluttered Scene	204
8.23	Efficiency ($\bar{E}_1$) vs. Reliability ($\bar{R}_1, \bar{R}_2$), (duck/6 DOF/cluttered) .	205
8.24	Efficiency ($\bar{E}_1$) vs. Reliability ($\bar{R}_1, \bar{R}_2$), (duck/6 DOF/cluttered) .	205

8.25 Noise vs. Reliability ($\bar{R}_1$)	207
8.26 Noise vs. Reliability ($\bar{R}_2$)	207
8.27 Boat, 4 dof, Occluded Scene	209
8.28 Parallel Implementation Pseudocode	214
8.29 Number of Processors (m) vs. Time (t [sec.])	215
C.1 Boat, 4 dof, Isolated Scene	242
C.2 Boat, 4 dof, Isolated Scene	243
C.3 Boat, 4 dof, Cluttered Scene	244
C.4 Boat, 4 dof, Cluttered Scene	245
C.5 Boat, 4 dof, Occluded Scene	246
C.6 Boat, 6 dof, Isolated Scene	247
C.7 Boat, 6 dof, Isolated Scene	248
C.8 Boat, 6 dof, Cluttered Scene	249
C.9 Boat, 6 dof, Cluttered Scene	250
C.10 Boat, 6 dof, Occluded Scene	251
C.11 Duck, 4 dof, Isolated Scene	252
C.12 Duck, 4 dof, Isolated Scene	253
C.13 Duck, 4 dof, Cluttered Scene	254
C.14 Duck, 4 dof, Cluttered Scene	255
C.15 Duck, 4 dof, Occluded Scene	256
C.16 Duck, 4 dof, Occluded Scene	257
C.17 Duck, 6 dof, Isolated Scene	258
C.18 Duck, 6 dof, Isolated Scene	259
C.19 Duck, 6 dof, Cluttered Scene	260
C.20 Duck, 6 dof, Cluttered Scene	261
C.21 Duck, 6 dof, Occluded Scene	262

Glossary of Symbols

p	point
M	model
I	image
S	scene
$\bar{\theta}$	pose
Θ	pose space
$M_{\bar{\theta}}$	model pose
s	surface
S	surface set
$\Gamma(,)$	projected surface
$\mathcal{A}()$	area constraint
$\mathcal{V}()$	volume constraint
ω	class
Ω	class set
v	voxel
V	voxel space
ρ	voxel resolution
ε	angle resolution
F	<i>free</i> voxel value
S	<i>surface</i> voxel value
O	<i>occluded</i> voxel value
U	<i>unknown</i> voxel value
t_v	stable voxel threshold
Λ	geometric probe function

$\lceil x \rceil$	ceil(x)
$\lfloor x \rfloor$	floor(x)
$\vec{r}$	ray
ϕ	frustum x-angle
ψ	frustum y-angle
k	filter threshold
$r(,)$	correlation function
r_t	correlation threshold
$\hat{r}$	estimated correlation
r^+	maximum correlation
$\hat{r}^+$	estimated maximum correlation
$\bar{r}^+$	average maximum correlation
T	tree
t	tree node
Φ	variable space
f	node decision rule
E	expected traversal cost
n_L	number of leaf nodes
n_I	number of internal nodes
$C()$	node evaluation cost
$\mathcal{W}()$	weighted node evaluation cost
π	tree path
n_π	path size
π	weighted tree path
$\mathcal{L}$	inclusion list
E_1	efficiency metric 1
$\bar{E}_1$	average E_1
$\bar{E}_{1p}$	pooled E_1
R_1	reliability metric 1
$\bar{R}_1$	average R_1
R_2	reliability metric 2
$\bar{R}_2$	average R_2
TSS	total sum of squares
SSB	sum of squares between
SSW	sum of squares within
LSD	Fisher's least significant difference

Chapter 1

Introduction

1.1 Object Recognition, Point Probe Strategies, and Dense Range Images

Object recognition is one of the oldest and most thoroughly investigated branches of Artificial Intelligence. There have recently been successful practical applications in specialized and constrained domains such as optical character recognition (OCR), fingerprint recognition, license plate reading, and face recognition. Despite approximately three decades of research, however, the realization of general and effective object recognition systems remains an elusive goal.

Feature based object recognition is one of the major paradigms in the literature. The central concept is that visual scenes contain common low level primitives or features, such as lines and corners. Subsequent to the extraction

of these features, recognition occurs in the space of feature relationships, rather than the much larger space of pixels.

Marr [1] was an early proponent of the use of low level features, which is attractive for a number of reasons. There is evidence that biological vision systems follow this approach, with features extracted at the retinal level and communicated to other brain regions for higher level processing. On a pragmatic level, feature extraction implicitly suggests a divide-and-conquer approach to partitioning object recognition into two smaller problems: first extract the features, and then match the arrangement of features against a database of (feature descriptions of) objects.

Feature extraction has proven to be a difficult problem in its own right. The visual world is not a line drawing, and identifiable edges do not readily occur in intensity images. Further, the more robust feature extraction methods tend to be elaborate and therefore computationally expensive.

There are some object recognition methods that do not follow the feature extraction paradigm, such as simple template matching, or its close relative the Hough Transform [2, 3]. Other examples include the Interpretation Tree [4] and Geometric Hashing [5, 6, 7], both of which can operate directly at the raw data level. It is interesting to note that following their exposition, each of these methods was modified to utilize higher level features [8, 9, 10].

A recent class of object recognition method which does not require feature

extraction preprocessing is *point probe strategies*. A point probe strategy is a minimalist technique whereby the object under investigation is probed in a structured sequence. The technique was initially motivated by robotic tactile sensing systems in which a robot-mounted tactile sensor repeatedly contacts specific points on the surface of the object. Given the time expense of the rather laborious collection of data, the motivation was to develop strategies which reduced the number of probes required to determine the object, with the position of each new probe being adaptively determined from the results of previous probes.

In the seminal paper of Cole and Yap [11], and in a majority of the subsequent works, the point probe strategy, which is more generally called *geometric probing*, was considered as a problem of computational geometry. The models were the class of planar convex polygons, and the scene under investigation consisted of a single isolated object with an unknown number of edges. It was assumed that the location of a single point was known which intersected either with the interior or a boundary (vertex or edge) of the model. The objective was to determine bounds on the necessary (i.e. lower bound) and/or sufficient (i.e. upper bound) number of probes required to determine and/or validate an object.

Cole and Yap defined a probe to be a directed line l , originating at infinity, which intersects the model M at a point p . The particular mechanism which

enacted the probe was not considered. In subsequent work [12], the probe mechanism was described as an omniscient *oracle* who dutifully and precisely communicates the point of intersection p of a queried l and M .

In practice oracles are difficult to come by and we are relegated to approximating their behavior with robot-mounted tactile probes or Coordinate Measuring Machines. Arkin et al. [13] have suggested that local operators such as gradient fields could be used in place of an oracle in visual data; however, the separation of figure from ground is a difficult problem. Indeed, Dobkin et al. [14] argue against the use of visual data in favor of simpler probe devices. One modality of visual data where the oracle assumption can be mapped fairly directly is binary vision, such as OCR.

Range images are another mode of visual data which is amenable to geometric probing. Unlike intensity image data, in which each pixel measures the intensity of light reflected from an area of the scene, range image data measure the distances along rays from the sensor origin to the scene's surface. In some cases, a range sensor can also measure the intensity of the reflected light from the intersected surface.

Once considered to be exotic, there is today a large number of range image acquisition technologies, most of which are commercially available. Laser based acquisition methods include triangulation, time of flight, depth-from-focus, and interferometry. An additional form of active ranging is structured

lighting, where a pattern of light is projected onto the scene and interpreted by an arrangement of intensity cameras. Stereography is a method of passive range sensing in which the 3D positions of natural scene textures are determined by triangulation from an arrangement of two or more intensity images. While the active ranging methods can generate dense images, the passive methods generate sparser point sets, the density of which is dependent upon the scene content.

Point probe strategies map nicely into the domain of dense range images, with the oracle replaced by a simple process of querying the range image at a specified point for the existence of surface data. Unlike the ideal case, where the oracle returns exact points of intersection, the use of real data introduces uncertainties which must be considered. Also, in more general versions of the object recognition problem, clutter and occlusions must be accommodated.

1.2 Brief Description of Method

In this thesis, a novel method for general object recognition is described based upon a point probe strategy in dense range images [15, 16, 17]. The problem domain which we address is much broader than has previously been considered within the point probe strategy literature. As described in the literature review of Section 2.1, with a few exceptions, previous works have

been restricted to the theoretical case of planar convex polygons and noiseless probes.

The problem which we address is general in the following aspects :

- **object shape:** We place no constraints on the shapes of objects, which may be freeform 3D surfaces. Indeed, the objects could even be composed of disjoint surface patches. The only restriction we assume is that the objects are rigid, and that their surface geometries are known apriori, i.e. as CAD or scanned models.
- **object position:** The objects can be positioned generally with 6 degrees-of-freedom (dofs) within the sensed scene. If the scene structure imposes some limitations on the possible pose of the object, then that information can be used to advantage.
- **scene content:** The imaged scenes may be cluttered, and the object of interest may even be partially occluded. The method can be extended directly to handle scenes containing multiple objects of interest.
- **image noise:** We handle real image data, and therefore must accommodate noise. We assume the most general *black box* noise model, i.e. that the noise occurs with some unknown distribution. When applied to a point probe strategy, this noise model translates to the *fallible ora-*

cle assumption, which admits that a query to the oracle is occasionally answered incorrectly.

The method is based upon an explicit representation of a binary point probe decision tree. Each internal node of the tree contains a reference to a single point, which is probed when the node is encountered during traversal. Depending upon the outcome of the probe, one of the two possible tree branches is followed, until eventually a leaf node is encountered. Each leaf node contains a list of all poses of the object which contain each of the surface points encountered while traversing the path from the tree root.

The construction of the decision tree representation involves generating and testing each possible model pose for the existence of the node points. Given the large number of possible model poses, tree construction can be computationally intensive. To improve efficiency, we quantize the image domain into a discrete voxel representation. This quantization has the added benefit of allowing us to deal with general shapes. Quantizing the image domain casts the point probe strategy into a form of efficient template matching. A dual problem to that which is addressed in this work is the efficient matching of a set of disparate templates.

1.3 Simple Example

As a simple example of the method, assume that we wish to recognize the L-shaped object in the 2D binary image illustrated in Figure 1.1, where the black *on* pixels are the surface of an object, and the white *off* pixels are the image background. The object has a fixed scale with respect to the image, and can be translated anywhere in the image with four possible orientations. The templates which the model orientations can create in the image are illustrated in Figure 1.2.

Let us randomly select image pixel p_1 and hypothesize that it lies on the surface of the object. This hypothesis will be true if there is any translation of the four allowable orientations of the model which intersects p_1 . We define an *interpretation* of the hypothesis as a translation of a model template such that some template pixel uniquely intersects with p_1 . As each of the 4 templates has exactly 4 pixels which can be translated to intersect with p_1 , there are 16 possible interpretations of the initial hypothesis.

Let us assume that each interpretation occurs with the same likelihood, i.e. there is no preferred orientation or translation of the model. We can determine the likelihood that a neighboring pixel of p_1 is *on* as the ratio of the number of times an interpretation intersects with the neighboring pixel over the total number of interpretations. This value is interpreted as the

Figure 1.1: 2D Binary Image

Figure 1.2: Model Templates

Figure 1.3: Likelihood of $p \in \mathbf{M}$ In Seed Neighborhood

likelihood that a pixel intersects with the surface of the model, given the truth of the initial hypothesis.

The set of all such likelihoods in the neighborhood of p_1 are illustrated in Figure 1.3. One possible strategy is to query the image at the next most likely *on* pixel. The result from this query will allow us to limit the number of possible interpretations. For example, if we query the image at pixel p_2 and find it is *off*, then the hypothesis can only be true for interpretations which intersect p_1 and do not intersect p_2 . The set of possible interpretations remaining after such a query outcome are illustrated in Figure 1.4.

A new set of likelihoods is constructed from this remaining set of interpretations, and again the most likely surface pixel is queried. The process

Figure 1.4: Remaining Interpretations

iterates in this manner until either a unique interpretation is found, thereby identifying and localizing the object, or all interpretations are ruled out as impossible, thereby refuting the hypothesis. If the hypothesis is refuted, then another image *on* pixel can be selected in place of p_1 , and the process repeats.

By using the knowledge of the object model to guide the sequence of image points to query, the object templates can be matched more efficiently than in a straightforward correlation. In the above example, a brute-force correlation of all templates would require between 4 and 16, and on average 10 pixel queries per image pixel. It can be shown that, following the proposed strategy, at most 5 queries per image seed are required. This difference in

efficiency becomes more pronounced for the recognition of more complicated models.

While the above example is simple, it will be shown that this basic approach to recognition extends directly to the use of 3D range data.

1.4 Thesis Claims

1.4.1 Major Claims

1. A novel method was developed for general object recognition in dense range images based upon point probe strategies.
2. The method was implemented and experimentally validated. The performance of various heuristic methods of decision tree design and tree traversal were empirically evaluated, under a variety of scene conditions. Experimental results indicate the suitability of certain combinations of tree design and traversal over others.
3. A new method of Geometric Probing is proposed. The *discrete point probe* queries the image at a specific point and returns one of a finite number of values which characterize the probed point. This probe method applies directly to dense range data.
4. Among the various alternative tree traversal methods, one method was

shown to be guaranteed to return the best solution, independent of tree design.

5. The relationship between decision tree point probe strategies and template matching was made explicit.
6. A method was developed that guaranteed matching for a set of disparate templates.

1.4.2 Minor Claims

1. A new definition of 3D object recognition is proposed as the solution of two constraint equations which measure the fit of two surfaces.
2. The computational complexity of various decision tree construction methods was analyzed, including a parallel method.
3. The computational complexity of various decision tree traversal methods was analyzed.
4. A statistical test was designed and applied for the selection of certain parameter values while controlling false positive and false negative recognition error rates.
5. A parallel method was developed for the generation of large decision trees.

6. The noise robustness of the object recognition method was demonstrated.
7. A parallel version of the object recognition method was developed, and its effectiveness was demonstrated.

1.5 Thesis Plan

In Chapter 2, a review is presented of the relevant literature in point probing, range image object recognition, and template matching.

In Chapter 3, a definition for 3D object recognition is proposed, and the basic recognition strategy is described as a sequential hypothesis testing method. Decision tree classification is introduced as an efficient method of classification, and the tree representation which we use is described. The recognition strategy is next shown to be equivalent to geometric probing.

Quantization is discussed in Chapter 4. A 3 parameter voxel space model is described, and a space carving method of image voxelation is presented. The effects of sampling and some practical techniques to filter noise are described. The duality between decision space and pose space quantization is explained in terms of voxel occupancy.

Chapter 5 deals with generating class templates from object surface models. A technique is described which uses random perturbations about a dis-

crete pose coordinate to determine the characteristic stable voxels of a class. An expression for the matching criterion, which is equivalent to the absolute sum of differences, is presented, and a simple experiment is reported which supports the discriminating power of this matching criterion.

Issues of decision tree construction are addressed in Chapter 6. An expression for the expected traversal cost is derived, and a number of construction methods are reviewed which are known to minimize the expected traversal cost. Optimal methods are argued as being too computationally expensive to be practical, and a number of cheaper sub-optimal methods are described. Four heuristic feature selection criteria are described, and their complexities compared. A practical technique for partitioning and parallelizing the tree construction method is then described, and some timing results are presented.

Chapter 7 presents a number of alternate tree traversal methods. One method is described as being guaranteed, in that it always returns the best matching solution. Four other heuristic methods, which are non guaranteed, are presented, and their complexity expressions compared.

The results of a suite of experiments are reported in Chapter 8. The experimental setup and performance metrics are described. A set of statistical tests are described and used to determine a suitable correlation threshold value. It is shown that significant differences exist between various combinations of tree traversal and tree design method. The efficiency vs. reliability

tradeoff is examined, and it is established that definite trends exist in the performances of tree traversal and design combinations. Certain of these combinations are shown to reside in a region of acceptably high efficiency and reliability. Further experiments are performed to establish the noise and occlusion robustness of the method, and a technique is proposed to enhance reliability. Finally, the time performance of a parallel implementation is examined.

In conclusion a brief discussion of research issues which directly follow from this work is presented in Chapter 9

Chapter 2

Literature Review

2.1 Point Probe Strategies

The probing problem was first formalized by Cole and Yap [11]. They defined a probe as a directed line, starting at infinity, which may intersect with the object at a point. In light of extensions to more general probe models the above probe is typically called a *finger probe*.

Cole and Yap described the use of probing strategies for *determination* and *verification*. In determination, also known as *reconstruction*, the objective was to exactly identify the probed object. In the somewhat simpler goal of verification the objective was to validate a hypothesized object identity. In both determination and verification the emphasis was on achieving the goal efficiently, with as few probes as possible.

There were three conditions which Cole and Yap assumed. The first

condition was that there was a single object under scrutiny. The second condition was that the probed object was a convex polygon and that the number of edges n was not known. The object resided in a 2D planar universe in the sense that all probing operations occurred in the plane of the polygon. The third condition was that there was a single *origin* point which was known to intersect the object either at its boundary (edge or vertex) or its interior. If this condition were not true, then an unbounded number of probes would be required for determination [12].

Having defined the problem, Cole and Yap then showed that $3n$ probes are sufficient and $3n - 1$ necessary for determination. In a much simpler proof, they showed that $2n$ probes are both necessary and sufficient for verification. Skiena [12] subsequently offered a simpler proof which showed that $3n - 1$ is a lower bound for determination.

In a refinement of the above problem, Berenstein [18] considered the case where the single object under scrutiny came from a finite set of polygons which could be preprocessed for greater efficiencies. He provided a strategy of ‘walking around’ the object and aiming probes for maximum discriminating effect. He proved the sufficiency of $2n + 2$ probes for determination. This result was particularly interesting in that it was dependent only on the number of sides of the polygon under scrutiny, and was independent of the number of objects in the set. Joseph and Skiena [19] improved upon Beren-

stein’s strategy and proved the necessity of $n - 1$ and the sufficiency of $n + 4$ probes for determination.

Lyons and Rappaport [20] considered the case of probing a finite set of m planar convex n -sided polygons with the additional conditions that each polygon be in a known position and orientation. They assumed that a vertex of each polygon in the set intersected at a common point, and that the edge adjacent to this vertex was collinear with a specified line. Given these additional restrictions, they showed the necessity and sufficiency of $m - 1$ probes for determination.

Alevizos, Boissonnat, and Yvinec [21] considered an unordered probe set and formulated a slightly different problem. They considered not only contact points resulting from probes, but also the freespace path that the probing device traversed in order to make contact. Assuming a knowledge of the probe paths, which they argue is a practical assumption for robot-mounted or other probe devices, they imposed the additional restriction that no probe path could intersect with a polygon edge.

Given the probed points and probe paths, their objective was to find a polygonal approximation (a *contour*) of the probed object such that the probed points formed the vertices of the contour and no probe paths intersected with the contour edges. They showed that the probe path information imposes a tight restriction on the interpretation of the points, and that when

such a contour exists, it is unique. They further proposed a method to obtain the contour in $O(n \log n)$ time.

The authors went further and applied this result to the determination problem [22]. They considered the case of a single possibly non-convex polygon of unknown n . There were two conditions that they assumed, the first of which was that of a known initial interior point. The second condition was that the object was a *simple* polygon, which meant that it had no collinear edges. Thus if the object edges were extended to infinite lines, the intersections of any two such lines would at most be a point.

A more powerful probe definition, which returned the normal as well as the point of contact, was admitted to accommodate the added complexity of probing a non-convex polygon. The authors claimed that without some additional information returned from each probe, determination for non-convex polygons is unbounded. They showed the necessity and sufficiency of $3n - 1$ probes for determination under this model. A method was also provided for planning each probe in $O(\log n)$ time, yielding an $O(n \log n)$ algorithm.

Biossonnat and Yvinec [23] extend their method to the case of multiple possibly non-convex polygons of unknown size. They use the additional information of the ray path to partition the scene into separate polygons. For a k polygon scene of n total edges, they show the sufficiency of $3n - 3 + k$ probes for determination.

Arkin et al. [13, 24] considered the case of probing a known finite set of k simple (possibly non-convex) planar polygons. They define an *arrangement* to be a set in which the model elements each have a fixed position and scale. The probed scene consisted of a single model drawn from the arrangement. This work was extended in [24], to deal with *concept classes*, an example of which is a set of various font renderings of the same letter.

They cast the point probe strategy within a decision tree formalism, (which was previously alluded to by Cole and Yap [11]), and considered two conditions on the arrangement. The first was that all models in the arrangement intersect at a single point, as was the case of Cole and Yap. The second condition was that the models in the arrangement were non-degenerate, meaning that the intersection of two edges at most formed a point (i.e. never a line).

Given the two conditions, it was demonstrated that there always exists a minimum height decision tree of height $\log k$. As will be discussed in detail in Chapter 6, minimum height decision trees are of interest because they are the most efficient to traverse given certain assumptions. It was demonstrated that the minimal height tree could be constructed in $O(n \log k)$ time for a total of n vertices in the arrangement, and that a lower bound for the construction time was $O(n + k \log k)$. Drawing on a result by Hyafil and Rivest [25] which concerned abstract decision trees, Arkin et al. further

proved that if either of the above two conditions were violated, then minimal height tree construction was NP-complete.

Given the difficulty of constructing optimal decision trees in the general case, they then considered a well known heuristic greedy strategy which is used to construct a balanced tree structure. They developed a bound of $O(hn^2)$ on the construction time, where h is the final height of the tree. They also proved that the greedy heuristic yields a tree that is at most a factor of $\log k$ times larger than the minimum height tree.

Lindebaum and Bruckstein [26] considered the problem of composite probing in which a number of probes are explored simultaneously. The problem was motivated by a K finger robotic hand sensing device. They dealt with the case of a single convex polygon with a known interior point, and developed lower bounds on the number of composite probes required as a function of K . They also developed strategies which solved the determination problem within a few composite probes of the optimal. They concluded that the use of composite probes reduced the number of probings required for determination by at most a factor of 3.

Ellis et al. [27] assumed the existence of an initial set of probe results, and proposed a representation to determine a probe which could contact a number of edges of possible interpretations. They considered a single object scene from a set of planar possibly non-convex polygons. The edges of each

model were projected into a parameter space such that the set of paths which could probe a given edge mapped to a trapezoid. The intersection of a number of such trapezoids represents the set of probes which could contact all corresponding edges.

All of the works described so far considered only the case of 2D planar polygons. Boissonnat and Yvinec [23] extended their 2D method to the 3D dimensional case to deal with 3D polyhedra. The polyhedra could be non-convex, but each polygonal face was restricted to be simple and convex. It was also assumed that an initial set of points were known, one for each polygon to be discovered. The main idea was to iteratively apply the strategy described in [23] for convex polygons. When exploring a given polygon, if a contacted point violated certain conditions, then it could not belong to the current polygon, and a separate process was invoked to explore the new region.

A sophisticated model of probing bounded polyhedra in any dimension of Euclidean space was presented by Dobkin, Edelsbrunner, and Yap [14]. They considered a number of different modes of probing and expressed the necessity and sufficiency of determination as a function of the number of vertices, edges, and faces, and the dimension of the space. In the same work, the author's analyzed the effects of uncertainty, such as noise. They derived a number of results for the case of finger probes of a convex planar polygon.

Owing to the subtlety of the argument, the author's admit that an extension to more general cases does not clearly follow.

In each of the above described works, the probing strategy was *adaptive*, in the sense that the results of previous probes were used to plan the placement of the next probe. A related problem is the interpretation of a set of existing probes, without any consideration of the order with which the probes were acquired. This problem was proposed by Grimson and Lozano-Perez [4] who considered the determination of a finite set of general 3D polyhedra from sparse 3D point data. They developed the interpretation tree approach which used features extracted from the data as constraints to prune the tree of possible interpretations. The interpretation tree approach has generated considerable attention in the literature, and is considered in greater detail in Section 2.2.1.2.

Belleville and Shermer [28] were concerned with the complexity of determining whether a given set of k probes were sufficient to discriminate a model. They considered the case of a finite set of convex planar polygons which were *strictly* convex, i.e. with no collinear adjacent edges. They required that the polygons be *unimodal*, a property which restricts a distance function from a vertex to all other vertices within a polygon to have a single maxima. They further restricted each polygon in the finite set to be in a known fixed position and orientation and to all intersect at a common point,

as in Lyons and Rappaport [20]. Given these conditions, they proved that, using two different types of finger probes, the problem of determining whether a given set of k probes are sufficient to discriminate a model is NP-complete.

While all of the above works consider the use of finger probes, other modes of geometry probe have been proposed. Some of these modes are abstract, while others correspond to actual sensing devices. Examples include:

- **hyperplane probe:** which measures the first contact between a d dimensional object and a $(d - 1)$ dimensional hyperplane moving parallel to itself ;
- **x-ray probe:** which measures the length of intersection between a line and an object ;
- **half-space probe:** which measures the volume of intersection between a half-space and an object ;
- **cut-set probe:** which measures the size of the cut-set of a graph and a specific partition of its vertices.

It is interesting to note that some of the above probe modes are quite similar. For example, an x-ray probe can be simulated with two finger probes. Also, there exists a duality between finger and 2D hyperplane probes (i.e. line

probes). This duality implies that results derived for one modality apply to the other, such as in [26].

2.2 Range Image Object Recognition

The earliest example of the use of range imagery for object recognition can be traced to Shirai and Suwa [29, 30], who stated that range data promised to be more reliable for object recognition applications than the more commonly used 2D intensity data. The recognition method itself was simplistic, consisting of comparing the shape and connectivity of planes extracted from the single object image to distinguish between a small set of polyhedra.

Grimson [31] has noted that the majority of methods formulate the object recognition problem as a search over one of two spaces. In correspondence space methods, the search space consists of all possible pairings between the model and image feature sets. In Pose Space methods, the search is applied over the space of positional parameters which transform the model coordinate reference frame into the image frame.

In all but the simplest cases, both of these spaces are too large to search exhaustively. Given m model features and s image features, the correspondence space is m^s large. If the object can be positioned freely in the scene, then the Pose Space is 6 dimensional. The problem is further complicated by

erroneous and missing scene data resulting from noise and occlusions, which may make an exact match impossible.

The universal approach to dealing with the exponential nature of either search space has been to identify and exploit whatever structure and constraints exist in the problem space, and to apply heuristics to direct the search. The main heuristic directs the search towards the most salient features, which are prominent and conspicuous features most likely to discriminate the object and most likely to be correctly extracted in preprocessing. The main structural property exploited by the majority of methods is what Faugeras and Hebert [32, 33] called the *rigidity constraint*. This property states that for rigid (i.e. non-articulated, non-deformable) objects the relative positions of the object's geometric parts are invariant under homogeneous transformations. Note that this is not generally true for the recognition of 3D objects in 2D intensity images, unless the scene is highly structured, such as when the object is constrained to lie on a plane in a known attitude.

The following review is a representative survey of the significant literature of object recognition methods using range images. For the sake of succinctness, we restrict the scope to methods which deal specifically with range images, noting any relation to more general 2D methods. The main classes of method are described, followed by specific examples. Where available, complexity measures and experimental results are described, including

time results for both feature extraction and matching.

2.2.1 Correspondence Space Methods

2.2.1.1 Attributed Graph

Attributed Graph methods compose both the model and scene features into separate graph structures where the nodes of the graphs represent features, and the edges some relationship between the features. The most common arrangement is with the nodes representing surface patches, and the edges representing adjacencies between surfaces [34, 35, 36]. Both the nodes and the edges are attributed with some descriptive fields, such as surface area for nodes, and solid angle between the normals of adjacent surfaces for edges.

Recognition consists of establishing an isomorphism between the scene and model graphs, which is an NP-complete problem. Further, due to noise, clutter and occlusions, as well as imperfections in the feature extraction processes, the problem is one of subgraph isomorphism, which is still more difficult.

The basic approach was first described for range image recognition by Oshima and Shirai [34]. Surface patch features were extracted from the range image and classified by analyzing the variance of the surface normal as either planar, one of 6 types of curved quadratic surfaces, or unknown.

Graph nodes were generated for each surface and were attributed with the surface area, perimeter, number of adjacencies, and other metrics. The graph edges were generated from the surface adjacencies, and were attributed with metrics such as the solid angle between the adjacent surfaces and the offset between the surface centroids.

Matching between the scene and model graphs was based on a heuristic which attempted to focus the search towards significant features which are the most likely to successfully discriminate the object. A kernel was selected from the scene graph which consisted of a set of two adjacent nodes called the *principle* and *subordinate* parts. The principle part was chosen as that node whose attributes maximized a heuristic function favoring large planar regions with many adjacencies, based upon the belief that such a large connected surface would have a greater likelihood of discriminating the object, and would be more reliably extracted from the image. The subordinate part was that surface which maximized the heuristic function among those surfaces adjacent to the principle part. The matching proceeded by searches through all of the model graphs, each of which represented a single object. A search tree was expanded for each candidate kernel match, and the adjacencies were evaluated. The search progressed in a mainly depth-first fashion growing the match out from the kernel. The search allowed some limited backtracking, with new kernels selected as required. Experimental tests were reported on

occluded scenes of regular objects, where a regular object was defined as one with a simple shape or obvious features. The feature extraction phase resulted in a small number of segments (approximately 10). The execution time of the feature extraction and matching phases were approximately 3 and 1 minutes, respectively.

In Fan et al. [35], rather than composing the graph from all model features, some of which would necessarily be occluded for a given pose, a number of graphs were generated for each model. For each characteristic model pose, only the visible quadratic surface patches of the model were used as nodes in the view-graphs. The nodes were attributed with pose dependent metrics in addition to shape information. Scene features were organized into a set of disjoint graphs, which roughly segmented the scene into separate objects.

Graph matching proceeded in three steps, with the first step comparing only the gross characteristics of the graphs, such as node count. The second step established node correspondences on those graphs which passed the first step, and calculated the transformations that mapped the model features into the image frame. The third step split and/or merged scene graphs to improve the matches. Experimental tests were performed on 32 views of each of 10 irregular objects, where an irregular object was defined as one with a free-form surface with no obvious features. The scenes contained 2 slightly occluded objects, with approximately 20 extracted features. The

feature extraction phase required 66 minutes of processing, and the matching executed in 3 minutes.

Kim and Kak [36] formalized the Attributed Graph method as a bipartite graph matching problem. A bipartite graph contains nodes that are separable into two disjoint sets (model and scene), and each edge connects a pair of nodes chosen from each set. Attributed graphs were generated for each model with the nodes representing planar and quadratic surfaces. The edges represented surface adjacencies which were attributed as one of four types of surface intersection. The matching occurred in 3 steps. The first step identified all models with a one-to-one injective mapping with the scene graph by applying 3 filters of increasing computational complexity, the final filter being a bipartite graph matching procedure. In the second step, those models which did not satisfy relational constraints were removed from further consideration in a process that used both bipartite graph matching and discrete relaxation. The final step determined the model pose and established exact correspondences, resolving any non-unique matches that may have resulted from the second step. Experimental tests were conducted on a database of 12 regular objects, each object having approximately 10 surface features. In scenes with isolated objects, the matching time was reported as 0.1 seconds, although no time results were reported for feature extraction or for scenes with occlusions. In both of the reported occluded scene tests,

of the approximately 10 scene objects only the 3 that were non-occluded were successfully recognized. A polynomial time implementation of the bipartite graph matching procedure was presented, although the complexity of the discrete relaxation process was not established.

2.2.1.2 Correspondence Tree

The Correspondence Tree approach organizes the search as a tree traversal, with each tree node representing a pairing between a model and a scene feature. A path through the tree represents a set of model-scene feature correspondences, called an *interpretation*, which ultimately disambiguates the object at a leaf node. This tree structure is a reflection of the exponential nature of correspondence space, and is far too large to explore exhaustively or optimally.

This approach can be implemented in either a data-driven or a model-driven fashion, with the tree nodes representing scene features in the data-driven mode, and model features in the model-driven mode. The main difference between the two variations is that the data-driven tree is represented implicitly as a by-product of the search, whereas the model-driven tree is precomputed offline and explicitly represented.

In the data-driven tree, the features that would describe the nodes are only known once the image has been acquired and the features extracted.

It is too costly to consider constructing the entire tree at that time, as the results of an object recognition process are in general desired as soon as possible after image acquisition. The complete data-driven tree can also be very space expensive, particularly when the scene is cluttered with many unknown objects. In the case of single object scene, however, Chen and Kak [37] argue that the search space of the data-driven approach will be smaller than the model-driven alternative.

In the model-driven approach, the node features are known independent of any specific image. The tree can therefore be constructed offline, and its structure modified to improve recognition performance. To reduce the space requirements, a separate tree can be constructed for each model of interest [38].

The most thoroughly explored method of this type is the Interpretation Tree of Grimson et al. [39, 4, 9, 40]. The tree is expanded during the search process by binding model and data features, one pair per node. The main idea of the method is to narrow the search by eliminating entire regions of the correspondence space that do not satisfy certain pose-invariant constraints. As new feature pairs are encountered, constraints are measured within sets of respective model and data features of near ancestor nodes and are compared. If these constraints are not satisfied, then the binding is not a valid possible interpretation, and is discluded. This effectively discards the subtree which

is rooted at this node, which can be a large subspace of interpretations.

The method was first proposed as a means of interpreting sparse range data, as might be acquired through a tactile sensor [39, 4]. The objects to be recognized were composed of 3D polyhedra confined to lie on a planar surface, and the data was processed in its raw form as simple 2D or 3D points with normals, requiring no feature extraction. Both binary (between two feature pairs) and tertiary constraints were considered. The binary constraints included the distance between two points, and the angle between two normals, and the tertiary constraints included the triple-product. Experimental tests on synthetic and real 2D and 3D images, as well as some real image tests, for isolated objects, yielded recognition times of about 5 seconds.

This initial work assumed that all of the sensed data points fell on some surface of the object, i.e. the scene contained a single isolated object. In [9], the method was extended to deal with cluttered and occluded scenes. Allowing that a sensed feature now might not correspond to any model feature, the concept of the NULL binding was introduced. The NULL binding was a wild card that signified that, in addition to the model features, a data feature may match some unknown extraneous feature.

It was found that this addition effectively increased the dimension of the search space and significantly increased its size. To counteract this, the use of higher level features, which were fewer in number, was favored. Rather

than raw point data, extracted edges were used in the 2D case, and planar surface patches in 3D. With the use of higher level features, it was possible to apply unary constraints to the bindings, which was similar to comparing node attributes in Attributed Graph methods. A number of heuristics were also applied to improve the search performance. It was concluded experimentally that the most significant advantage was from the use of a Hough Transform preprocessing stage which segmented the scene into clusters of features likely to fall on single object. Tests were conducted on cluttered and occluded range images of polyhedral objects with 6 dofs. The feature extraction time was approximately 1.5 minutes, and the recognition time 30 seconds.

The performance of the Interpretation Tree method has been theoretically analyzed by Grimson in a number of publications. In [40], the problem was described as one of consistent labeling. Considering the case of point data and isolated scenes, he presented a combinatorial argument to derive bounds for the expected number of sensory points required to disambiguate a model. In [31], the time complexity of the method was derived. For isolated scenes, the search complexity was determined to be quadratic in the number of model features, and quadratic in the number of data features. In a cluttered or occluded scene, the search complexity was exponential in the size of the correct interpretation.

The geometric constraints compared between sets of feature pairs by

Grimson were pose independent, and the model pose was not explicitly calculated until after the tree expansion. Faugeras and Hebert [32, 33] recognized that a unique pose could be calculated from a small number of feature bindings, usually 2 or 3, depending upon the feature type. They called this property the *rigidity constraint*, and used it to apply both data- and model-driven strategies to limit the depth of search tree expansion. As soon as the tree was expanded in a data-driven mode to the minimum required depth, the model transformation was calculated and applied to all model features for model-driven verification against scene features. The features used were planar and quadratic surface patches, curved lines where the surface undergoes C1 discontinuity, and points of extreme curvature. In experimentation on a scene with an isolated free-form object, matching was achieved in 5 to 30 seconds. Feature extraction times were not reported.

The 3D-POLY system of Chen and Kak [37] also used a variant of the rigidity constraint in a two-phase tree expansion. In the first phase, a hypothesis of the model pose was generated by identifying a set of scene features of minimal size necessary to constrain the objects pose. They labeled this the *hypothesis generation feature set*, or HGF. Hypothesis generation was equated to expanding the search tree to level h , which was equal to the cardinality of the HGF. The hypothesis was verified by transforming the m model features by the computed pose, and expanding the remaining

$m - h$ tree levels. Prior to tree expansion, the model features were compiled into a feature sphere structure based upon their principle directions for efficient indexing. The features used were surface patches (planar, cylindrical, conic), edges (straight, ellipsoidal), and extremal points (vertices, curvature extrema). Experimental results were reported for occluded scenes of 2 regular objects, with approximately 100 scene features. The performance of the feature extraction was not reported, and the matching was approximately 4 to 10 seconds. The time complexity of the method was derived as $O(m^2)$ for the case of an isolated object.

Flynn and Jain [41] presented the BONSAI system as a fairly straightforward implementation of the Interpretation Tree approach. Planar, cylindrical, and spherical surface features were used, and the constraints were described as unary and binary predicates. The breadth of the search tree was initially limited by filtering the model database, and only considering those models with features similar to the extracted scene features. The depth of the search tree was limited by verifying an interpretation as soon as the feature bindings allowed the calculation of unique pose estimate, as in Chen and Kak's method [37]. Experimental tests were reported for 100 synthetic and 10 real images of isolated and cluttered scenes. The model database contained 20 regular models consisting entirely of planar, cylindrical, and spherical surfaces. The feature extraction time was not reported, and the matching time

varied widely from a few seconds to hours. The authors explained the large matching times by claiming that efficiency was not an explicit design criterion in the experimental testbed. The objects were correctly classified in the synthetic images in 95% of cases, with a 60% to 70% correct classification rate in real images.

The Local Focus Feature method (LFF) of Bolles and Horaud [42, 43] was an example of an early model-based method where the tree was not pre-computed and explicitly represented. The main heuristic in LFF was to immediately limit the breadth of the search by identifying distinctive features in the scene, and then use the model to direct the matching of additional features. In this way, the search space was limited to the subset of image features related to the initial matched focus feature, which was similar to the use of a kernel to identify graph isomorphisms in Oshima and Shirai's Attributed Graph method [34].

The selection of an initial focus feature was a heuristic function of uniqueness (i.e. how well a feature discriminated the object), utility (how much new information it added), extraction cost, and the likelihood of detection. Once selected, a maximal clique algorithm was applied to identify the largest set of correspondences relating to the focus feature. The method was first developed for 2D applications [43], and later extended to 3D [42]. The 3D implementation used only edge features in the matching stage, which were

believed by the authors to be more descriptive than surface features. Planar patches were used for verification. Experimental results were reported for an occluded scene of a single regular model with circular edges, although neither algorithmic complexity nor time performances were reported, and it was not clear for the simplistic cases described that a brute force approach would not have sufficed.

Ikeuchi [44, 45] presented a model-driven method in which the tree was explicitly precomputed offline. The method borrowed from aspect graph methods, which generate representations of a model viewed from a collection of vantage points. Ikeuchi partitioned all aspects of the model into attitude groups, which are sets of vantage points with the same visible features and topology. The search tree was then constructed with the leaf nodes representing a unique attitude group, and the intermediate nodes representing a single feature common to the descendant leaf attitude groups. In constructing the tree, the constraints applied to each node were selected manually. During recognition, the processing began by focusing on the highest point in the range image, with features extracted only around this region of interest. The search was a simple depth-first traversal. Experimental results were reported on an occluded scene for a single regular model with 7 attitude groups. No feature extraction or recognition times were reported, and only those objects that were non-occluded or slightly occluded were successfully recognized.

A main objective of the CAGD system of Hansen and Henderson [38] was to automatically generate a strategy tree from geometric models. First, *level 1* features were identified by applying a set of filters that measured heuristic qualities such as rare, robust, cost, completeness, and consistency. A tessellated viewing sphere was used as an accumulator of the number of different aspects from which any feature was visible. If no single feature was observable from all aspects, a spanning set of features was used at the root of a tree. Each level 1 feature set had an associated tree, called the *corroborating evidence subtree*, which contained additional features that could be used to verify the initial match and constrain the object's pose. The features used were based upon a modified winged-edge representation, and tests were reported on a simple isolated polyhedron. No algorithmic complexity or time performance were presented, although the authors reported that feature extraction was significantly slower than matching.

2.2.1.3 Alignment

The Alignment method of Huttenlocher and Ullman [46, 47] was also based on the principle that only a small number of feature correspondences are needed to constrain the object's pose. All possible sets of correspondences of this minimal size were generated and then tested by computing the transformation and mapping all of the remaining model features into the scene.

Each match was scored simply by comparing the transform model feature locations against the image feature locations. This method was almost identical to the earlier RANSAC method of Fischler and Bolles [48], with the exception that RANSAC emphasized stochastic rather than exhaustive selection of test cases.

Formulated in this way, the size of the search space was low order polynomial, which is much smaller than the exponential size of the general correspondence space. For m model and s scene features, the search space was $O(m^k n^k)$, where k typically equals 2 or 3, depending upon the feature type and the dimension of the object's pose space.

Alignment was described first in [46] for 2D images, with the objects allowed all 6 positional dofs (called the *3D from 2D* problem). Edge features were used, described as corner and inflection points. As soon as an image feature was matched, it was removed from further consideration, which accelerated subsequent matchings. The features were ordered heuristically to aid the search, with features on the same edge, and those close to the center of the image, being considered first.

2.2.2 Pose Space Methods

2.2.2.1 Generalized Hough Transform

The Hough Transform is one of the oldest and most thoroughly investigated image processing methods [49]. Also known as pose clustering, transformation clustering, and hypothesis accumulation. it is a useful technique for both feature extraction and complete object recognition in 2D and 3D images.

The central concept is that each image feature is independently mapped into a discrete parameter space. Each such mapping casts a vote for a particular parameter value, or a family of values. All votes are accumulated in the parameter space, and high values indicate that a number of features share the same parameterization, which is evidence that they lie on the same object.

The Hough Transform simplifies the recognition problem to one of peak detection in an accumulator array. It benefits from a degree of noise and occlusion immunity. As each feature is dealt with independently, it is also inherently parallelizable. A disadvantage of this class of method is that the parameter space dimensionality may be too high for a straightforward implementation to be practical. A high dimensional parameter space not only makes the storage expensive, but also makes the generation of parameter space hypersurfaces and peak detection computationally costly. Another

disadvantage is that a different parameter space is in general required for each feature class.

A further disadvantage is that the discretization of the parameter space can lead to inaccurate results. Grimson and Huttenlocher [50, 31] presented a theoretical derivation based upon combinatorics and concluded that the discrete nature of the parameter space can introduce ambiguities in the peak detection phase. They concluded that moderate amounts of scene noise, occlusion, and clutter can dramatically reduce the effectiveness of this method class. In other work [9], however, Grimson and Lozano-Pérez conclude that a coarse Hough is a useful preprocessing step for clustering image features into objects, which can then be more efficiently recognized in an Interpretation Tree based search.

Hough initially described the method in a patent for the detection of straight lines and circular arcs in 2D binary and intensity images [51]. The work was subsequently introduced to the image processing community by Duda and Hart [52]. A variant of the method called the Generalized Hough Transform was proposed by Ballard [53] to detect arbitrary shapes. Stockman and Agrawala [54] noted an equivalence between the Hough transform and template matching.

Muller and Mohr [2] described a Hough method for the detection of planar and quadratic surfaces in 3D range images. Separate parameter spaces

based upon the implicit surface equations were used for each surface type, resulting in 3 to 7 dimensional spaces. To deal with this high dimensionality, a hierarchical method based upon iterative splitting was used to reduce the representation cost. A neighborhood operation was applied to each range point to estimate the surface parameters.

Henderson and Fai [8] assumed that planes were first extracted from the range image, and presented a Hough method to detect polyhedra. The 3D parameter space was in this case not a pose space, but rather was based upon intersecting spheres in R^3 . For a model polyhedron, a reference point was selected so that its distance to each plane vertex was unique. This set of unique distances was called the *shape model*. For each scene vertex, every cell was incremented which fell within a sphere centered at the vertex, and with a radius equal to a distance from the shape model. The maxima of the accumulator values were then candidates for the polyhedra reference point.

Krishnapuram and Casasent [3] extended the 2D straightline Hough to detect planes and polyhedra in range images. This method acted directly on the range data points and did not require any feature extraction preprocessing. The parameterization was based upon the property that the dot product of a plane normal and a vector from the coordinate frame origin to any point on a plane yields a constant value. A 3D parameter space was composed of the plane orientation values and the normal distance of the plane to the

origin. Transforming the planes by translation, rotation, or scaling, affected the mapping to parameter space in a known way. Based upon this, a hierarchical algorithm was developed to decouple the transformation parameters for a polyhedron. Experiments on 3 simple isolated polyhedron in simulated range images required a recognition time of approximately 10 minutes.

2.2.2.2 Geometric Hashing

Geometric Hashing is similar to the Hough Transform in that scene features are mapped into and accumulate votes within a discrete parameter space. Unlike Hough methods, the parameter space necessarily encodes a spatial relationship between model features which is invariant under the allowable object transformations. For example, if the object can be rotated in the scene, then any object rotation will lead to the calculation of the same hash table entries.

In the preprocessing phase, a model-centered coordinate reference frame is defined by selecting a small set of model features, called the basis feature set. The cardinality of this set is small (typically 3), and depends upon the allowable model transformations and the feature geometry. The coordinates of all other model features are calculated with respect to this reference frame, and an entry is added to the appropriate parameter space bin. This entry references the identity of the model, the basis feature set, and the encoded

feature in model coordinates. This process is repeated for all possible model basis feature sets.

In the recognition phase, all scene features are parameterized with respect to a basis set composed of randomly selected scene features, and the respective hash table entries are incremented. If all of the features in the basis set fall on a model, then a significant number of the incremented parameter values will reference this model and basis set, which is then passed to a verification phase.

The main advantage of geometric hashing is its efficiency. Any number of models can be encoded in the hash table without significantly affecting the recognition execution time, although the size of the hash table increases with each additional model. The algorithmic complexity of the recognition phase is $O(s^{k+1})$ for s scene features and basis set cardinality k . Additional strengths of the method are robustness to noise and occlusions, and the applicability of parallel implementations.

Also known as Affine Invariant Matching, the method was first proposed by Lamdan et al. [5, 7] for recognition in 2D images. The features used were points which exhibited either sharp concavities or deep convexities, called interest points. The allowed transformations were affine, which describe the projection of generally positioned 3D object geometry to the 2D image plane. The affine mapping required a basis set of cardinality $k = 3$, selected from

non-collinear interest points. It was shown that, for s interest points, the worst case time complexity of the recognition phase was $O(s^4)$. If the first 3 features fell on the object, then the best case complexity of $O(s^2)$ was realized. Good results were obtained for essentially binary images of two overlapping (occluded) household items, with no other scene objects. In [6], this method was extended to deal with line and convex and concave curve features.

Cohen and Wolfson [55] described a geometric hashing method in the 3D domain, where the features were provided by a CAD/CAM system. The feature locations and geometries, which included points, lines, planes, some quadratic surfaces, as well as some solids, were known exactly from the system database, and required no sensing component.

Barequet and Sharir [56] described a voting method inspired by hashing. The recognition problem was formulated as the matching of two 3D point sets A and B , which describe the same surface (in part), and which can be relatively rotated and translated. The method was based upon the observation that for a point $\vec{p} \in A$, the Euclidean distances $|\vec{p} - \vec{q}|$ for all $\vec{q} \in B$ map into a rotated and scaled image of A in a voting table. The scale factor of the mapped image was $2\sin\frac{\Theta}{2}$ for a rotation Θ , and so the voting table entries were more compact for a correct value of Θ . The complete algorithm used a two-phase binary search through the 3D space of rotations, regenerating and

evaluating the dispersion of the voting space at each iteration. Experimental results found the acceptable matches between images of approximately 2,000 points in approximately 30 minutes. The method could be applied to point sets describing volumes as well as surfaces, and required no feature extraction.

2.2.3 Recent Methods

Two recent methods are noteworthy in that they both address the same generality of object recognition problem as this work. There exist some algorithmic similarities.

The spin image method of Johnson and Hebert [57, 58] characterized the surface surrounding each surface point of the object with a type of 2D template. A 2D coordinate system was defined at each point. One dimension was the radial distance from the normal of the oriented point, and the second dimension was the signed elevation from the plane defined by the point and its normal.

The spin image was a discrete representation of this coordinate frame. For each vertex on the triangular mesh model's surface, each other vertex was mapped into a discrete coordinate, or bin. The bin value was incremented with each mapped element, so that the spin image acted as a type of histogram. One nice characteristic of the spin image is that it is invariant

to translations and rotations.

The models were preprocessed offline, with the spin images generated and stored for each model vertex. During recognition, spin images were generated for each image location and matched against the database using what was essentially template matching. If a match was detected, an iterative closest point routine was invoked to improve the fit. The spin image method has been successfully demonstrated on database of 20 objects, some of which were free-form surfaces.

In [59], the method was improved to enhance efficiency for a multi-object database. The size of each spin image was compressed by an approximate factor of 10 using a principle component analysis. It was shown that recognition reliability degrades gracefully with the degree of compression, and that recognition efficiency was improved, although time results were not reported.

The point signature method of Chua and Jarvis [60] is also based upon a view independent characterization of each surface point. A point signature is generated at a model surface point by computing the intersection of a sphere of a specified radius with the model. A one dimensional signal is composed from the distances of the points of intersection with the sphere. The signal is offset in a repeatable way by zeroing its largest component.

During preprocessing, these signatures are generated for a collection of points on the surface of each model. During recognition, the point signatures

are extracted from the image and matched against the database using simple threshold comparisons. Because the signatures are succinct, the matching is very efficient. Experimental results on cluttered and occluded scenes of multiple objects demonstrate successful recognition in tens of seconds.

2.3 Template Matching

Template matching is one of oldest methods of 2D binary and intensity image processing. A template is a subimage which is representative of some pattern of interest. Matching a template in an image is equivalent to finding those image locations where the pattern represented by the template occurs. In cases where a meaningful template can be defined, template matching is attractive due to its simplicity and robustness.

The main limitation of template matching is its expense. The brute-force exhaustive method is to translate the template to all possible image locations, and compare it with the overlayed image window. The comparison is done by calculating a similarity metric, such as the sum of absolute differences, or the cross correlation. For m template elements and n image elements, the exhaustive approach requires $m^2(n-m+1)^2$ comparison operations. Much of the literature on template matching has been concerned with devising ways to reduce this computational expense.

2.3.1 Single Template Matching

Much of the classical literature on efficient template matching deals with the case of matching a single template against an image. Nagel and Rosenfeld [61] reduced the expense of template matching by ordering the sequence in which the template and image pixel pairs were compared. Those template pixels with a high expected difference from a randomly selected image pixel were matched first. This ordering increased the likelihood that an error threshold, which signified that the template and image were not matched at the location, was exceeded prior to comparing all pairs.

Margalit and Rosenfeld [62] analyzed the ordered search method and modeled the performance improvement probabilistically. They described how a template pixel ordering and a rejection threshold could be determined based on an analysis of the image and template histograms. They presented examples of expected performance for classes of histogram distributions, and explained the experimental results with respect to the presented probabilistic model. They also discussed the merits of applying the ordered search method to processed versions of the template and image, called local property arrays.

A similar but more general approach was presented by Barnea and Silverman [63] who described a class of algorithms called Sequential Similarity Detection Algorithms (SSDA). They presented examples where template and

image pixel pairs were selected randomly and compared against a threshold value. Cross-correlation was used as a similarity metric, and a full template comparison was performed only at those locations where the metric exceeded some threshold. There were a number of techniques proposed to select the threshold value, such as the constant threshold variant (which was the most similar to [61]), and the monotonic increasing threshold (in which the threshold value increased as a function of the number of pixel pairs tested). The SSDA approach was more general than the ordered search of Nagel and Rosenfeld, in that the order of the template locations within the image was also considered. Experimental results showed a 50 times speed up in the registration of satellite images.

Munteanu [64] evaluated the use of SSDA on binary images. He compared the results of two sub-optimal thresholding sequences, and analytically explained measured performance differences. The small performance differences encountered suggested that an optimal thresholding sequence may not lead to much improvement over the suboptimal sequences.

Another approach to improve performance of template matching was to organize the matching process hierarchically. A succession of comparisons were made between less detailed, more efficient representations of either or both of the template and image. The simplest hierarchy had just two levels of detail. Vanderbrug and Rosenfeld [65] described a two-stage matching

approach where, in the first stage, a subtemplate which was a subset of the pixels from the initial template was matched at each image location. Only those locations with a high enough score were further processed in second stage, which consisted of a full template match. The computational cost of the two stage approach was equal to $A + B \times C$, where :

- A = the cost of applying the subtemplate,
- B = the probability that the subtemplate match will succeed,
- C = the cost of applying the full template.

Increasing the size of the subtemplate had the effect of increasing A and decreasing B . This cost formula was optimized to derive an expression for the optimal (least cost) template size which was solved analytically for the binary case.

Rosenfeld and Vanderbrug also investigated the use of resolution coarsening in two stage template matching [66]. Rather than a subtemplate, a coarse resolution template was used in the first processing stage. The template and image were divided into larger blocks over which the average was calculated. The same cost expression described in [65] was optimized for this case.

In both subtemplate and reduced resolution cases, the objective of the first processing stage was to efficiently identify image locations which were

most likely to match the full template. In Margalit and Rosenfeld [67], the use of extracted feature probabilities was proposed to this same end. In the first stage, features which were common in the template, and rare in the image, were extracted from the image. They proposed a probabilistic model which used prior knowledge about the distribution of features in the image to find likely locations for template matches. For this approach to be useful, it was noted that the feature extraction process itself need be efficient.

In other work, more than two levels of hierarchy were employed. Wong and Hall [68] applied the SSDA approach to multiple levels of resolution. They calculated a number of low resolution versions of both the image and template. Each level had its own ordering algorithm, and for a given level, only those locations where the threshold was exceeded would be processed in subsequent levels. They theoretically derived an expression for computational efficiency as a function of the number of levels, the window size for averaging, the image size, and the scene content (represented by a constant factor). Experimental results were presented on satellite images using both constant and monotonic increasing threshold functions. The results demonstrated efficiency improvements over exhaustive matching on the order of 10^3 .

Tanimoto [69] organized the image and template hierarchy in a pyramid structure. A pyramid is a stack of reduced resolution (averaged) images. The

bottom of the pyramid was the original image, and the top a single pixel, representing the average of the entire image pixel values. Each level had $1/4$ the number of pixels of the next higher resolution level. Template matching was achieved using a recursive descent algorithm. Tests were performed with a variety of descent policies (i.e. which descendants of a cell were to be searched), sensitivity policies (i.e. what thresholds were to be used at each level), and termination policies (i.e. when to stop).

In the above works, an assumption was made that the template was a subimage which was close enough to the imaged object that it scored identifiably when overlayed at the correct position. In certain applications, an initial position estimate of the object is known, but the exact shape is not. An example is medical imagery, where the rough location of structures are known, but their shapes may vary due to anatomical variances. The Deformable Template method (also known as Flexible Templates and Active Templates) has been developed to address this class of problems, and consists of three components :

- a template with control parameters which vary its shape and/or position,
- an objective function which measures the similarity of an instance of the parameterized template to an image,

- a search method to optimize the objective function.

Lipson et al. [70] applied Deformable Templates to the localization of vertebral trabecular bone in 2D Computed Tomography images. They used an elliptical template, with five parameters representing the origin, major and minor axis, and the rotation of the ellipse. The method first performed an edge extraction on the intensity image, producing a binary edge image. A gradient field was then created from the binary edge image, so that the pixels in close proximity to the original edges had the highest field value. The template parameters were varied over 40 iterations to maximize a correlation value. A second phase was applied next which used a shallower edge field and only 15 iterations. Experimental results were generated from a 552 image set, representing 69 patients with 8 images per patient. The method proved successful for all but two of these trials.

Hill et al. [71] recognize that the parameter search space was very large, and developed a Genetic Algorithm based search to address the problem. They applied their method to echocardiogram data (2D intensity ultrasound images of the heart). The template had 10 parameters, and was created by manually extracting control points from a training set of 66 representative images. Experimental tests on 20 images compared well with manual operation.

In [72], a Simulated Annealing based search was used for the recognition of traffic signs in 2D intensity street images for automatic guided vehicle navigation. Unlike the medical applications, the location of the object in the image was not initially known. Their template model had 5 parameters, two of which were positional. Experiments recognized traffic signs in typical images in approximately 10 seconds. By comparison, the exhaustive matching method executed on one of the test images for 10 hours.

2.3.2 Template Set Matching

All of the works cited in the previous section addressed the efficiency of matching a single template against a single image. Due to the low descriptive power of any single template, it is desirable to be able to efficiently match a number of templates to a single image. Tanimoto [69] recognized this, and stated that methods which increase the efficiency of a single template match would aid in increasing the efficiency of multiple template matching.

Ramapriyan [73] directly addressed the issue of efficient template set matching. The set of templates were organized into a tree structure, where the leafs of the tree corresponded to the individual templates. The intermediate nodes corresponded to a single representative template (RT), which was the union of all descendant templates. Matching began at the root of the tree, and as the tree was traversed, the node whose RT best matched

the image location was expanded further. For a given tree, Ramapriyan analytically determined values for the search order, the threshold sequence, and the sequential similarity function for each RT. He did not describe how to compose the tree, i.e. how to decide the number of levels, the partitions of the template set, and the composition of the RTs. Experimental results on synthetic images showed a performance improvement factor of 4 for a set of 36 templates.

2.3.3 3D Template Matching

The previous 2 sections have described methods which apply template matching to 2D binary and intensity image data. Template matching has also been applied to 3D range imagery. Bhattacharya and Qian [74] encoded a binary voxel representation of a model as a polynomial, and show how template matching could be performed on the polynomial encoding. In addition to template matching, the execution of other operations was presented, such as rotation, scaling, morphological filtering, and shape decomposition. The emphasis of this work was on the representation formalism, and efficiency and implementation issues were not addressed.

Newman et al. [75] applied template matching to the verification step of automated industrial inspection. Industrial inspection is similar to the medical imaging applications in that the identity and (nominal) pose of the

object are known. The purpose was to verify that both the position and shape of the object were correct, so the part could proceed to subsequent manufacturing stages.

The range image was organized as a $2\frac{1}{2}$ D representation (depth map on x-y grid). This representation of range imagery is particularly suitable for template matching, as it is a direct extension of the standard 2D intensity image representations, albeit with each pixel encoding a depth rather than an intensity value. As the position of the object was constrained, there was no search component to identify the template location in the image, and a straightforward similarity measure was applied between the acquired image and a synthetic image generated from an idealized model. To speed up the template matching, only subtemplates were generated from the model. Experimentation showed that just 1% of model datum points needed to be compared for robust results which allowed 5 degrees in orientation and 0.25 inches in translation positional inaccuracy. The execution time was approximately 1 sec on SPARCstation 2.

In [76], the authors extended this work to allow some unknown (although constrained) positional parameters. The range image was a composite of multiple images of the same object, taken from different vantages. A method called bidirectional template matching was implemented. First a subtemplate of the model was generated and compared to the image, following which a

subtemplate of the image was generated and compared to the simulated model data. Prior to the template matching phase, the unknown positional parameters (3 translational and 1 rotational) were resolved using a silhouette method.

In addition to inspection applications, template matching has been proposed for recognition within range imagery. Dawson and Vernon [77] describe the use of template matching on secondary representations derived from range image data. The secondary representations they used were 1D functions of orientation histograms, depth maps, and needle maps. They generated these secondary representations for all possible orientations of a given model, and successfully recognized a number of industrial parts. The use of template matching in secondary representations is similar to template matching within local property arrays [62]. Snell et al. [78] also applied Flexible Templates to the surface segmentation of brain structures in 3D MRI images.

2.4 Summary

2.4.1 Point Probe Strategies

The literature on Geometric Probing has evolved within the framework of Computational Geometry. Most of the work subsequently deals with a some-

what idealized version of problem, i.e. the case of a single isolated planar convex polygon without noise. The majority of the work are concerned with analyses of the complexity of the probing problem. There are no references in the theoretical literature to the particular probe model presented in this work (i.e. the binary point probe).

Despite the richness of the theoretical work, there are very few reported applications. There are some robot-mounted tactile probe systems, which closely follow the premises of the theoretical probe model. There has been no applications using dense range data.

2.4.2 Range Image Object Recognition

A large amount of effort has been directed towards the classical 3D object recognition problem, and it remains an active area of research. Two main themes which recur throughout the literature are the use of the rigidity constraint and feature saliencies to facilitate search through the exponentially large spaces. Despite the heavy reliance on features, the particular features used for most methods are chosen in an *ad hoc* fashion that is rarely justified. In every case where the execution times were reported, and independent of the types of features used, the time expense of the feature extraction processing was greater than that of the subsequent matching.

There were some exceptions to the rule, where features were not extracted

from the scene prior to matching. The early Interpretation Tree work [39, 4] used sparse 3D point data, as may be acquired directly through tactile or range sensing. It was eventually concluded, however, that higher level features were necessary for reasonable performance in cluttered and occluded scenes [9]. The trend was similar for Geometric Hashing methods, which was initially formulated for point data and extended to higher level features [5, 6, 7, 10]. The 3D Hough methods of Muller and Mohr [2] and Krishnapuram and Casasent [3] were purely point oriented, although Henderson and Fai [8] assumed extracted planes.

Despite the large body of literature on the subject, there is very little common ground for quantitative comparison between object recognition methods. Experimental results are difficult to compare, due to differences in the experimental setup, including the types of objects supported (polyhedron, freeform surface, etc.), the degrees of positional freedom, the number of different objects in the database, and the scene complexity (isolated, cluttered, or occluded). The time results are difficult to compare directly, as the computational platforms differ, and indeed span a number of generations. Further, the analysis of computational complexity, which is the most theoretical and fundamental way to compare the performance of methods, has been reported only spuriously in the literature. It may also be misleading, as the expressions are often based upon assumptions about features, which

may depend heavily on the performance of the feature extraction routines, as well as the scene content.

2.4.3 Template Matching

The classical literature on template matching addressed the problem of the efficient matching of a single template. There has been very little work on matching template sets.

An exception is the work of Ramapriyan, who describes a tree structure which is similar to that presented in this work, in that individual templates make up the tree leaf nodes. A difference is that, in our work, intermediate sample tree nodes represent the intersection, rather than the union, of the descendant node templates. This difference is subtle but important, as it allows for a much deeper tree structure, which is necessary in handling a large template set. A union of a sufficiently large template set would likely be very similar to that of any other large set, unless the members are carefully selected to be extremely similar within a set and dissimilar across sets.

Another difference is that Ramapriyan's method required a full template correlation evaluation with the RT at each visited tree node, whereas the described method more efficiently evaluates only the single node element value. In this way, each sample tree node voxel is like a single element sub-template of all descendant templates.

Ramapriyan also did not relate his template tree to decision tree classification, which has a larger and formal literature.

Chapter 3

Description of Method

There have been many general and informal definitions of Machine Vision. As with most processes close to our everyday experience, formal definitions are more rare. One formal definition by Grimson casts object recognition as a problem of consistent labeling of image and scene features [40, pg. 668]. The dependence of this definition on extracted features makes it inappropriate for this work, and motivates the following definition.

3.1 Recognition as a Constraint Satisfaction Problem

Let a scene S consist of a set of s surfaces $\{s_j\}_1^s$ in $\mathcal{R}^3$. Let a range image I of S be a set of n points $\{p_i = (x_i, y_i, z_i)\}_1^n$ which sample the s_j in S , with the point coordinates described in the sensor coordinate reference frame.

In keeping with most range data acquisition technologies, we assume that any $p_i \in \mathbf{I}$ correspond only to portions of the surfaces in $\mathbf{S}$ which are non-occluded with respect to the sensor vantage. An occlusion occurs when some portion of a surface s_a lies between the sensor head and some other surface portion s_b , so that only those $p_i \notin s_b$ are measured. Two types of occlusion are illustrated in Figure 3.1. In *self-occlusion*, some portion of the object surfaces are hidden from view by other sections of the surfaces on the same object. In *external occlusion*, an object is (partially) hidden behind some other object.

Figure 3.1: Types of Occlusion

Let $\mathbf{M}$ be a surface model of a rigid object, which may be transformed rigidly within $\mathcal{R}^3$ to any 6 dimensional pose $\vec{\theta}$. We denote an instance of $\mathbf{M}$ with pose $\vec{\theta}$ as $M_{\vec{\theta}}$. The objective of object recognition is to determine all occurrences of $M_{\vec{\theta}}$ in the subspace of $\mathbf{S}$ that is imaged by $\mathbf{I}$.

Let $\Gamma(M_{\vec{\theta}}, \mathbf{S})$ be the set of surfaces defined by the projection of the non-occluded surfaces of $M_{\vec{\theta}}$ onto the closest corresponding surfaces of $\mathbf{S}$. For every non-occluded $p_m \in M_{\vec{\theta}}$, there exists a corresponding $p_g \in \Gamma(M_{\vec{\theta}}, \mathbf{S})$ such that $p_g \in \mathbf{S}$ and $|p_m - p_g| \leq |p_m - p_s| \forall p_s \in \mathbf{S}$.

One way to measure the goodness of a match between $M_{\vec{\theta}}$ and $\mathbf{S}$ is to consider the magnitude of the area of $\Gamma(M_{\vec{\theta}}, \mathbf{S})$ and the magnitude of the volume bound by $M_{\vec{\theta}}$ and $\Gamma(M_{\vec{\theta}}, \mathbf{S})$. In the case of a perfect match, for example, the area of $\Gamma(M_{\vec{\theta}}, \mathbf{S})$ would equal that of $M_{\vec{\theta}}$ and the volume between them would have a magnitude of zero.

Let $\mathcal{A}(\mathbf{S}_i)$ denote the magnitude of the area of a set of surfaces $\mathbf{S}_i$ and let $\mathcal{V}(\mathbf{S}_i, \mathbf{S}_j)$ denote the magnitude of the minimal volume bound by two surface sets. A 2D example of these constraints is illustrated in Figure 3.2. In this reduced dimension example, length corresponds to area, and area corresponds to volume.

Figure 3.2: Constraints

Object recognition can then be defined as a process of finding all values of $\vec{\theta}$ which satisfy the following constraints :

$$|\mathcal{A}(M_{\vec{\theta}}) - \mathcal{A}(\Gamma(M_{\vec{\theta}}, \mathbf{S}))| < \varepsilon_{\mathcal{A}} \quad (3.1)$$

$$\mathcal{V}(M_{\vec{\theta}}, \Gamma(M_{\vec{\theta}}, \mathbf{S})) < \varepsilon_{\mathcal{V}} \quad (3.2)$$

where $\varepsilon_{\mathcal{A}}$ and $\varepsilon_{\mathcal{V}}$ are thresholds determined by the accuracy of the sensor and model, and the noise and occlusion properties of the image and scene. These constraints are similar to the metric that is minimized in iterative closest point (ICP) algorithms which are commonly used in pose refinement processes [79].

These constraints form a necessary but not sufficient condition for recog-

dition : all correct solutions will satisfy the constraints, but some incorrect solutions may as well. There may be arbitrary scene surfaces which satisfy the constraints, but are not part of $\mathbf{M}$. There may also be scene objects which are similar in shape to $\mathbf{M}$, and satisfy the constraints in certain poses.

In addition to the purely geometric information considered in equations 3.1 and 3.2, we could include constraints which measure reflectance properties of the object, such as color and texture. While this is possible within the described recognition framework, properties other than surface geometry are not explored within the scope of this work.

3.2 Matching by Sequential Hypothesis Testing

The objective of a recognition system is to generate a set of $M_{\vec{\theta}}$ which satisfy some matching criteria, given an image and a model database. In our case, the constraint functions define the matching criteria. The straightforward exhaustive method of generating matches would be to reorient the models to all possible poses and evaluate the constraint functions.

While the exhaustive method would certainly be reliable and identify all possible matches, it is too costly to be considered practical. For example, assume that the 3 orientation components of an object's 6 dimensional pose

space were quantized fairly coarsely into 10 degree units, and that the object could be translated within the image by 100 increments along each of the 3 translational axes. The exhaustive method would then require $\approx 5 \times 10^{10}$ constraint evaluations per model.

At the heart of any object recognition method is some way of determining matches without testing all possibilities. Our method is based upon a sequential hypothesis testing strategy which iteratively reduces the number of possible matches. The strategy hypothesizes the intersection of the object with some selected image point, and searches for additional surface data at locations relative to that point. At each increment, the image is queried for the existence of surface data at a specific spatial location, and the set of possible object poses is further restricted. Eventually, either the object is identified and localized, or the initial hypothesis is refuted.

Let Θ be the pose space of $\mathbf{M}$ in $\mathbf{S}$, which is the complete space of poses with which the model can possibly be positioned in the scene. If there are no restrictions, then the most general Θ will be 6 dimensional, with 3 orientation and 3 translation parameters. If there is some additional scene structure, such as fixtures or a table-top environment, then Θ can be of lower dimensionality.

Let Θ_i be a subspace of Θ . The set of all instances of $\mathbf{M}$ with some pose

$\vec{\theta}$ in Θ_i is denoted as M_{Θ_i} :

$$M_{\Theta_i} = \bigcup_{\vec{\theta} \in \Theta_i} M_{\vec{\theta}} \quad (3.3)$$

The sequential hypothesis test starts with the random selection of a range data point p_0 in $\mathbf{I}$. We formulate the initial hypothesis that p_0 lies on the surface of the model ;

$$H_0 : p_0 \in \mathbf{M} \quad (3.4)$$

Note that H_0 makes no assumptions about the pose of $\mathbf{M}$. Indeed, at this stage the hypothesis does not establish correspondence between p_0 and any particular surface point of $\mathbf{M}$. For H_0 to be true, p_0 can be any point on the surface of $\mathbf{M}$. The truth of H_0 therefore restricts the possible poses of $\mathbf{M}$ to the subset Θ_0 which contains only those $\vec{\theta}$ such that $M_{\vec{\theta}}$ intersects with p_0 :

$$H_0 : p_0 \in M_{\vec{\theta}} \quad \forall \quad \vec{\theta} \in \Theta_0 \quad (3.5)$$

Conversely, if H_0 is false, then the only possible $\vec{\theta}$ are those for which $M_{\vec{\theta}}$ does not intersect with p_0 :

$$\overline{H_0} : p_0 \notin M_{\vec{\theta}} \quad \forall \quad \vec{\theta} \in \Theta_0 \quad (3.6)$$

At the next stage, the hypothesis is extended by querying $\mathbf{I}$ at another point p_1 in M_{Θ_0} . If surface data exists in $\mathbf{I}$ at p_1 (denoted $p_1 \in \mathbf{I}$) then we

form the hypothesis which asserts that an instance of $\mathbf{M}$ exists in some pose such that both p_1 and p_0 intersect it's surface. Alternately, if no surface data exists at p_1 , then we hypothesize that p_0 falls on the model surface and p_1 does not.

$$H_1 = \begin{cases} p_1 \in \mathbf{M}, p_0 \in \mathbf{M} & \text{if } p_1 \in \mathbf{I} \\ p_1 \notin \mathbf{M}, p_0 \in \mathbf{M} & \text{if } p_1 \notin \mathbf{I} \end{cases} \quad (3.7)$$

We respectively call these the *true* and *false* stage 1 alternative hypotheses, and denote them by H_1^T and H_1^F . We continue extending the hypothesis by querying $\mathbf{I}$ at specific points relative to the initial seed point. In general, at any stage i , one of the two alternatives is extended:

$$H_i = \begin{cases} H_i^T : p_i \in \mathbf{M}, H_{i-1} & \text{if } p_i \in \mathbf{I} \\ H_i^F : p_i \notin \mathbf{M}, H_{i-1} & \text{if } p_i \notin \mathbf{I} \end{cases} \quad (3.8)$$

At each stage, the size of the pose subspace is reduced until there are eventually only a small number of $M_{\hat{\theta}}$ that can satisfy the hypothesis, which are subsequently verified.

While the stated objective of recognition is to find matches defined by constraint functions, the sequential hypothesis testing strategy makes no direct reference to these constraints. The strategy is based on establishing potential correspondences between model and image surface points which, if correct, would necessarily satisfy the constraints of equations 3.1 and 3.2.

Figure 3.3: Probe Order

3.3 Query Point Sequence

In the previous section, the criteria with which specific query points were selected to extend the hypothesis was not considered. The order of the query point sequence has an impact on performance and will be addressed in Chapter 6, which describes the tree construction process.

An example of the query point ordering is illustrated in Figure 3.3, which shows a portion of a range image of a duck object against a planar back-

ground. The *voxel* representing the image seed is rendered in blue, and labeled 0. The points are labeled according to the sequence with which they were interrogated. Points which fell on a surface, and therefore resulted in an extension of the *true* alternative hypothesis, are colored green. Points which did not fall on a surface, resulting in a *false* alternative hypothesis extension, are colored red.

3.4 Classification

The sequential hypothesis testing strategy can be described as a multi-stage classification process. In a classification system, the characteristics of an input signal $\vec{x} = (x_1, x_2, \dots, x_n)$ are compared with those of a set of predetermined patterns, which are called classes and denoted as ω_i . The result of a classification is the labeling of $\vec{x}$ as belonging to one (or possibly many, or none) of the ω_i . Each variable x_i of $\vec{x}$ can take on a value from a range if it is continuous, or one of k fixed values if it is k -ary discrete.

We consider each x_i to form an orthogonal axis in an n -dimensional *decision space*, denoted as $\mathcal{D}^n$. If the variable values come directly from a measurement, then $\mathcal{D}^n$ is an *observation space*. Alternately, if the variables consist of features extracted from a measurement, then $\mathcal{D}^n$ is a *feature space*. Note that the term *feature* has a more general connotation than its previ-

ous use in the introductory chapter, and (unless otherwise indicated by the context) the current definition will apply to all subsequent use within this document.

The ω_i are regions in the decision space, also known as *decision regions*. The borders of these regions are called *decision boundaries*, and are usually $n - 1$ dimensional surfaces.

The main issues in designing a classification system are :

1. **Feature Extraction** : In most cases, $\mathcal{D}^n$ is a feature space, so an effective mapping from observation space to feature space must be determined. The advantage of feature extraction is that the resulting feature space is of lower dimension than the initial observation space [80, 81]. The use of features can also focus the classification to those characteristics of the signal which are most discriminating.
2. **Training** : In a *supervised* classification system, the classes are known apriori, and the decision boundaries result directly from the choice of decision space. In an *unsupervised* system, the distinction between classes is not known apriori, and so the decision boundaries must be determined from data in a process called *training*.
3. **Classification** : In all except the simplest cases, there will be some alternatives in determining how to label an input signal. The decision

Figure 3.4: Decision Space

regions may not be disjoint, in which case $\vec{x}$ may fall in more than one ω_i . For example, in Figure 3.4 classes ω_4 and ω_5 intersect, and any $\vec{x}$ which falls in this intersection could be labeled as either class. Alternately, $\vec{x}$ may not fall within any ω_i and the class label will have to be determined based upon some measure of closeness to a decision boundary. The efficiency of a classification process may also be a concern, particularly when the number of classes or the dimension of $\mathcal{D}^n$ is large.

4. **Evaluation** : Some measures of the effectiveness of the designed classification system, such as the misclassification rates, must be determined.

Of the above four issues, we are primarily concerned with the final two, as we do not extract features, and our classes are predefined from the object model.

3.4.1 Decision Tree Classification

One way to classify an input signal is to compare its proximity to all classes by calculating some distance metric. This is called *single-stage* or *one-shot* classification and is equivalent to the exhaustive method described above.

In cases where either the number of classes or the dimension of the decision space is large, a *multi-stage* classifier can be a much more efficient alternative. At each stage of multi-stage classification, only a small subspace of the decision space is considered, often consisting of only a single variable. Based upon the value of the input signal within this subspace, the number of potential class labels is restricted, thereby further reducing the computational burden in subsequent stages.

A *decision tree* is one way to represent a multi-stage classifier. Each node t_i of a decision tree has associated with it a *class set* Ω_i , with Ω_0 of the root node t_0 containing all classes. For a k -ary tree, each *non-terminal* node has descending from it between 2 and k *child* nodes. In addition, each non-terminal node has associated with it :

- a *variable space* Φ : a subspace of the complete decision space,

- a *decision rule* f : a discrete function which evaluates the Φ of $\vec{x}$ and returns the identity of a child node.

As the tree is traversed, the decision rule of each encountered non-terminal node is invoked and the identified child node is followed. When a *terminal* (or *leaf*) node t_k is encountered, the input signal is labeled as belonging to one of the classes in Ω_k , often subject to further verification.

An example of a decision tree classifier for the decision space of Figure 3.4 is illustrated in Figure 3.5. Assuming that all ω_i are equally probable, and that the expense c of evaluating either variable is identical, then the expected cost of classification $E(C)$ is the average tree path length multiplied by the variable evaluation cost, which is $2.4c$. Even in this simple case, $E(C)$ is less than that of single-stage classification, which is $10c$.

Decision trees have been well investigated in the literature. A survey is presented by Moret [82], and the subject is thoroughly treated by Breiman et al. [83]. A formal definition of a decision tree is offered by Moret et al. [84, pg. 581], which closely follows Knuth's definition of a ordered trees [85, pg. 305], [82]. The great interest in decision trees has been due in part to their generality. They have been applied towards the solution of a broad range of problems (not all of which are classification problems) which include :

- binary function evaluation

Figure 3.5: Decision Tree

- diagnostic problems (e.g. medical diagnosis, machine diagnosis)
- decision table programming
- switching theory
- analysis of algorithms
- template matching
- machine vision

The main advantage of decision trees are their efficiency : the expected cost of traversing the tree is a logarithmic function of the total number of

classes. One disadvantage of decision tree classifiers is reliability, as they cannot in practice guarantee to operate correctly or completely. Kurzyński [86] derived a decision rule which minimizes the probability of misclassification, given a tree structure and feature selection, and a complete knowledge of the feature probability distributions and class prior probabilities. The tree traversal policy was, however, essentially a branch-and-bound search, which was too computationally expensive to be practical in most cases.

Another disadvantage of decision trees is the large amount of computation required to construct even sub-optimal trees.

3.5 Sample Tree

The *sample tree* is a decision tree classifier based upon the above described hypothesis testing strategy. Each node of the sample tree represents an alternative hypothesis at a single stage of the sequential hypothesis test. The node variable set is the point that is tested at each extension, and all possible alternative hypotheses are pre-compiled and stored in the tree structure.

The sample tree derives its name from the observation that each decision rule interrogates 3D space for the existence of range data. The tree therefore encodes a strategy or schedule for image sampling.

3.5.1 Structure

The sample tree has a binary structure, with each non-terminal node t_i having exactly 2 children nodes. The variable space Φ_i for each t_i is a single point p_i , and the decision rule f_i determines whether or not p_i lies on some scene surface, i.e. whether $p_i \in \mathbf{I}$.

The two children nodes of t_i partition Ω_i into two disjoint sets, with the *true* child containing those $\omega_j \in \Omega_i$ that contain surface data at p_i , and the *false* child containing those which do not. The *true* and *false* child nodes therefore represent H_{i+1}^T and H_{i+1}^F , respectively.

3.5.2 Decision Space

The decision space of the sample tree comprises a binary variable $\mathbf{x}_i$ for every point $p_i \in \mathcal{R}^3$. The value of $\mathbf{x}_i$ indicates the presence or absence of surface data in the scene at p_i . This decision space is infinite dimensional, which poses a number of potential difficulties. As described in Chapter 5, domain quantization resolves these difficulties by reducing the dimension of the decision space to a finite and manageable, although still large, size.

Each class corresponds to a specific model pose. Thus, if class ω_j corresponds to $M_{\vec{\theta}}$, then

$$p_i \in \omega_j \longleftrightarrow p_i \in M_{\vec{\theta}} \quad (3.9)$$

There exists a separate class for every pose within the pose space of the model. As with the dimensionality of the decision space, we will see that the infinite number of classes becomes manageable through domain quantization.

There are two conditions which further restrict the class definition. First, as each class represents a simulated range image of the corresponding $M_{\vec{\theta}}$, only the non-occluded surface points are included in the class. Second, classes are constructed only for those $M_{\vec{\theta}}$ whose non-occluded surface intersects with the decision space origin. The rationale for this restriction is explained in the description of the decision tree traversal which follows.

3.5.3 Traversal

We allow scene clutter, so that an image may contain many objects, some of which are unknown. We therefore do not wish to recognize the entire image as one object or another. Rather, we would like the classifications to apply to subimages, with, for example, some part of the image labeled as class ω_a , another part as class ω_b , etc.

Object recognition in cluttered scenes is a more difficult problem than that of the single isolated object case. In the Interpretation Tree method, for example, an expression for the time complexity of the tree search was derived to be quadratic in the number of data and model features. Modifications to

effectively deal with cluttered and occluded scenes, however, increased the search complexity to an expression exponential in the size of the correct interpretation [87].

Rather than segmenting the image, which is the traditional means of separating an image into its parts, we traverse the sample tree repeatedly, classifying different regions of the image. Traversing the tree is equivalent to applying a sequential hypothesis test with each image point in turn as p_0 . An input signal for classification is created from an image $\mathbf{I}$ by translating $\mathbf{I}$ so that some surface point $p_s \in \mathbf{I}$ intersects with the origin of the decision space. We call p_s the *seed point*, and generate a different input signal with each image point in turn as the seed. This is equivalent to classifying the neighborhood of every image point.

3.6 Relation to Geometric Probing

The described sequential hypothesis test qualifies as a form of geometric probing, which are known to be representable as decision trees [11, 13]. The probe model used differs from other types described in the literature [12], and follows as a direct consequence of the use of dense range image data.

In the finger probe model of Cole and Yap [11], a probe is defined to be a directed line l , and a probe outcome as the point p of contact between l and

the boundary of a scene object. This probe model Λ_f can be expressed as a function which queries the scene with the specified probe and returns the probe outcome ;

$$p = \Lambda_f(l) \quad (3.10)$$

Note that the specific scene queried is implicit to this function, and that the probe outcome may be the point at infinity.

All other geometric probe models can be cast in a similar expression. We define a *discrete point probe* model Λ_{dp} , where probe is a point p and the probe outcome is a discrete symbolic value k ;

$$k = \Lambda_{dp}(p) \quad (3.11)$$

In our case, there are only two possible outcomes to each probe, making it a binary point probe model.

3.7 Block Diagram

A process flow block diagram of the complete recognition method is illustrated in Figure 3.6. This diagram includes offline preprocessing as well as online recognition processing stages. In the experimentation, we are concerned mainly with the results of the tree traversal and template matching processing steps (i.e. the phase 1 and phase 2 results). The subsequent verification stages are included in the diagram for completeness.

Figure 3.6: Process Flow Block Diagram

3.8 Summary of Chapter

In this chapter, some basic concepts and definitions were introduced. The 3D object recognition problem was defined as the satisfaction of two constraint equations which measured the alignment of two 3D surfaces.

As an efficient alternative to exhaustive matching, a sequential hypothesis testing procedure was proposed. The characteristics of classifications systems in general and decision tree classifiers in particular were presented. It was shown that the sequential hypothesis testing procedure could be represented as a decision tree classifier, which we called the *sample tree*. Aspects of the structure, decision space, and traversal of the sample tree were discussed.

It was described that the sample tree encodes a geometric probe strategy. The particular probe model used was, the *discrete point probe* model, was described.

Finally, a block diagram of the complete recognition process was presented.

Chapter 4

Quantization

The convenient description of the sample tree decision space comprises an orthogonal axis for every point in $\mathcal{R}^3$. This results in an infinite dimensional decision space, which poses a number of practical difficulties, such as efficient representation and manipulation.

To circumvent these difficulties we quantize the decision space. Quantization is a process of approximating a continuous signal with a discrete signal from a finite set of values [88]. In our case, the domain of the signal under approximation is $\mathcal{R}^3$, and each continuous point in $\mathcal{R}^3$ maps to a discrete point in a discrete space. We call such discrete points *voxels* and the discrete space a *voxel space*, denoted respectively as v and V . The term *voxel* is a concatenation of *volume* and *element*, and is the 3D analogy of the pixel (picture-element).

Each voxel represents a small unique cube of space, and the voxel space

is a collection of these cubes over a region of $\mathcal{R}^3$. A voxel space encodes a uniform quantization of $\mathcal{R}^3$, with each voxel of identical size. An example of a non-uniform quantization of $\mathcal{R}^3$ is the *octree*, in which voxels are grouped hierarchically into a series of larger cubic regions and organized as a tree structure. The octree is considered to be a higher level and sometimes more efficient organization of a voxel space.

In discretizing $\mathcal{R}^3$, we are coarsening the spatial resolution of the image to a level that is appropriate for the task at hand. Spatial resolution coarsening is a well established technique in both 2D and 3D image processing. The two main benefits of resolution coarsening are succinctly summarized by Ballard and Brown [89, pg. 106] :

First, the computations are fewer because of the reduction in dimensionality. Second, confusing details present at the high resolution version of the image may not appear at the reduced resolution.

Examples of 2D image processing methods that use resolution coarsening are coarse-fine template matching [66] and hierarchical structures such as image pyramids [69] and quadtrees [90], which are the 2D equivalent of octrees. In 3D image processing, voxels and octrees have been used to advantage in a number of applications, including mesh generation [91, 92], image

registration [93], and recently recognition [94]. Voxels have also formed the basis of some 3D graphics rendering methods [95]. For example, Goldwasser et al. [96] developed a computing architecture called the *voxel machine* for the display and manipulation of medical volumetric data such as CAT scan images.

In some range image processing methods discrete structures other than voxels are used, often to represent domains other than the measurement space. For example, Johnson and Hebert [57] proposed a method of mapping a range image into a set of discrete 2D histograms. These histograms were called *spin-images*, and were invariant to rotations about a normal vector defined at specific image points. This rotational invariance lowered the dimension of the search space during matching. In the *point signature* method of Chua and Jarvis [60], a one dimensional signal was constructed by sampling the intersection of the range image with a sphere centered at an image point. The intersection of the image with the sphere was called a *space curve*, and was sampled at discrete angular increments of 15 degrees, resulting in a signal of 24 values. Binning methods, such as the Hough transform [49] and Geometric Hashing [7] also make use of discrete histograms as a main representation.

There are also many methods that do not explicitly map into a discrete domain, but do allow some error envelope on the interpretation of the value of

a continuous variable, which effectively quantizes the domain of the variable. For example, in the Interpretation Tree method of Grimson et al. [4], the constraints used to compare image and model features were assigned error tolerances in translational and rotational dimensions.

In our method, the decision space was quantized into a discrete voxel space, which was used to represent the range image data. The quantization of the decision space imposes an effective quantization on the model pose space, limiting the number of model poses that are observable at the discrete image resolution.

4.1 Decision Space Quantization

The discrete representation of the decision space is a voxel space $\mathbf{V}$ which is fixed to the coordinate reference frame of the range sensor and contains its scanning frustum. There are three parameters which completely describe a (uniform) voxel space mapping: *resolution*, *bounds*, and *offset*.

The *resolution* parameter measures the granularity of the quantization as the length of each voxel along a given axis, denoted as ρ_x , ρ_y , and ρ_z . In our usage, and typically, resolution is of uniform length in all dimensions, i.e. $\rho_x = \rho_y = \rho_z = \rho$. The volume of a voxel is therefore ρ^3 .

The *bounds* parameter limits the region of $\mathcal{R}^3$ over which $\mathbf{V}$ is defined.

The bounds can be described by the minimum and maximum corners of $\mathbf{V}$, denoted respectively as $p_- = (x_-, y_-, z_-)$ and $p_+ = (x_+, y_+, z_+)$.

Given its resolution and bounds, the size $n_{\mathbf{V}}$ of a voxel space, i.e. the total number of voxels in $\mathbf{V}$, is calculated as :

$$n_{\mathbf{V}} = \lceil (x_+ - x_-)/\rho_x \rceil \times \lceil (y_+ - y_-)/\rho_y \rceil \times \lceil (z_+ - z_-)/\rho_z \rceil \quad (4.1)$$

where the delimiters $\lceil \cdot \rceil$ round their argument up to the nearest integer.

The *offset* parameter determines the location of the voxels in $\mathcal{R}^3$. The offset is denoted as δ , and takes a value in the range $[0, \rho]$. There can be a separate offset value for each voxel space dimension, respectively denoted δ_x , δ_y , and δ_z . One option is to set the offset value so that all points in the region $(\pm \frac{\rho_x}{2}, \pm \frac{\rho_y}{2}, \pm \frac{\rho_z}{2})$ map to an origin-centered voxel. Another possibility is to have the $\mathcal{R}^3$ origin lie on the border of more than one voxel, in which case a point in the neighborhood of the origin could map to at most 8 distinct voxels. As an example of the effect of the offset parameter, consider the quantization $u(\cdot)$ of a one dimensional signal x :

$$u(x) = \lfloor \frac{x+\delta}{\rho} \rfloor, \delta \in [0, \rho] \quad (4.2)$$

where the delimiters $\lfloor \cdot \rfloor$ round their argument down to the nearest integer.

This mapping is illustrated in Figure 4.1 for various values of δ .

Figure 4.1: Quantization Function

A function $\mathbf{V}()$ which maps point $p = (x, y, z)$ to voxel $v = (i, j, k)$, $i, j, k \in \mathcal{I}$ is a simple extension of the above one dimensional case :

$$\mathbf{V}(p) = (i, j, k) = \left(\left\lfloor \frac{x - x_- + \delta_x}{\rho_x} \right\rfloor, \left\lfloor \frac{y - y_- + \delta_y}{\rho_y} \right\rfloor, \left\lfloor \frac{z - z_- + \delta_z}{\rho_z} \right\rfloor \right) \quad (4.3)$$

This calculation is very efficient, which is one of the benefits of the voxel space mapping. In our case, we set all offsets equal to half of the resolution (i.e. $\delta_x = \delta_y = \delta_z = \delta = \rho/2$) so that there exists a single voxel centered at the voxel space origin.

One way to represent a voxel space is as a 3D array in which case the voxel coordinates (i, j, k) act as array indices. If ρ is small and the bounds of $\mathbf{V}$ are large, then n_v may be too large to explicitly represent and store each

$v \in \mathbf{V}$. In cases where only a small percentage of the voxels are significant, an option is to represent $\mathbf{V}$ as a hash table, with entries only for significant voxels [92]. For example, if only surface rather than complete volumetric information is important, then the size of the hash table will be related to the surface area of the represented scene, rather than the volume of $\mathbf{V}$.

4.1.1 Image Voxelation

The process of mapping a range image $\mathbf{I}$ into a voxel space $\mathbf{V}$ is called *image voxelation*, and is denoted $\mathbf{V}(\mathbf{I})$. The goal of image voxelation is to generate a voxel representation of the scene surfaces which are sampled by the range image. The result of image voxelation is that the scene is characterized by assigning each voxel in the sensor's field-of-view one of the four following values :

- **surface (S)** : the voxel intersects some image surface data,
- **free (F)** : the voxel lies between the sensor origin and a surface-valued voxel,
- **occluded (O)** : the voxel lies behind a surface-valued voxel with respect to the sensor origin,
- **unknown (U)** : the voxel lies outside of the sensing frustum.

Space carving is a method of image voxelation which is similar to ray tracing methods in computer graphics. The space carving principle is based upon the observation that, for a point p to be sensed on a scene surface, the space between p and the sensor origin must be empty. Each p therefore contains information not only about it's surface of intersection, but also about the ray of freespace between p and the sensor origin.

Image voxelation proceeds by first setting all voxel values to $\mathbf{U}$. For each range datum p_i , the value of the voxel $\mathbf{V}(p_i)$ which intersects p_i is set to $\mathbf{S}$. This indicates that the voxel $\mathbf{V}(p_i)$ intersects with some scene surface. The equation is then generated for a ray $\tilde{\mathbf{r}}_i$ which starts at the sensor origin and passes through p_i . This ray is the path through space that the laser light traveled when p_i was sensed during image acquisition. Unless other information indicates the contrary, it can be asserted that any voxel which lies between the sensor head and p_i and intersects $\tilde{\mathbf{r}}_i$ does not intersect with a scene surface. Starting at the sensor origin and extending to a boundary of $\mathbf{V}$, each $\tilde{\mathbf{r}}_i$ is sampled at a specified increment $\Delta\tilde{\mathbf{r}}_i$. The magnitude of $\Delta\tilde{\mathbf{r}}_i$, denoted $|\Delta\tilde{\mathbf{r}}_i|$, is some fraction of the voxel space resolution ρ :

$$\Delta\tilde{\mathbf{r}}_i = \frac{|\Delta\tilde{\mathbf{r}}_i| \tilde{\mathbf{r}}_i}{|\tilde{\mathbf{r}}_i|}, \quad 0 < |\Delta\tilde{\mathbf{r}}_i| < \rho \quad (4.4)$$

The ray sampling produces for each $\vec{r}_i$ a sequence of $n_{\vec{r}_i}$ points :

$$\{p_{\vec{r}_i,j} = \vec{r}_i + j\Delta\vec{r}_i\}, j = 1 \dots n_{\vec{r}_i} \quad (4.5)$$

Visiting each point in order, the values of the voxels containing each point are tested and possibly changed. If the value of $V(p_{\vec{r}_i,j})$ is equal to U or O, then $V(p_{\vec{r}_i,j})$ is assigned the value F. Otherwise, if the value of $V(p_{\vec{r}_i,j})$ equals S, then some previously treated image points fell within the voxel, and the value of $V(p_{\vec{r}_i,j})$ remains unchanged. Once an S-valued voxel has been encountered within a ray point sequence, then all subsequently encountered U-valued voxels in the sequence are set to O.

An example of the results of image voxelation for a single plane of light is illustrated in Figure 4.2.

4.1.2 Sampling Effects

There are two sampling effects which can introduce errors in the image voxelation process. The first stems from the angular separations $\Delta\phi$ and $\Delta\psi$ between adjacent rays, and places a restriction on the voxel resolution ρ . Undersampling will occur if the angular separation between adjacent rays is so large that two adjacent points on a well-behaved sampled surface are separated by a distance greater than ρ . This would result in the interspersion of U-valued voxels among the voxelated surfaces. Undersampling errors

Figure 4.2: Image Voxelation

can be avoided by setting the value of ρ to be greater than the maximum distance separating two adjacent points. An upper estimate of this distance is defined by the separation of the two points formed by the intersection of two adjacent rays with the far field-of-view plane of the sensing frustum.

The second sampling effect results from the incremental sampling along a given ray, and occurs when a ray $\vec{r}_i$ passes through a voxel v very close to a corner. It may happen that, even though $\vec{r}_i$ intersects v , none of the points in the sequence $\{p_{\vec{r}_i,j}\}$ fall within v , so that the value of v erroneously remains unchanged. The frequency of this error can be reduced, at an increased computational expense, by decreasing the value of $|\Delta\vec{r}_i|$. The error rate cannot,

however, be completely eliminated as a ray can always pass arbitrarily close to a voxel corner. A sufficiently small value of $|\Delta\vec{r}_i|$ will, however, remove all occurrences of this error that are of practical concern while incurring a reasonable computational expense.

4.1.3 Noise Filtering

An additional source of error affecting image voxelation is image noise, which is not caused by the voxelation process but can nevertheless be effectively filtered in the discrete domain. A simple model treats the image noise as high frequencies in the range image, which can be removed by rank filtering. Rank filtering will also remove legitimate high frequency components of the scene. This is deemed acceptable as one of the stated objectives of resolution coarsening is the removal of unnecessary detail from the image.

A rank filter which uses the voxel space as a histogram is applied to the voxelated image. Initially, all voxels are set to the integer value zero. Each image point is then indexed into the voxel space in turn, and the value of each referenced voxel is incremented. Each voxel therefore contains a count of the number of image points with which it intersects. Only those voxels with a sufficiently high number of intersecting points are set to the value S .

For a given voxel v the point count threshold to be exceeded is a function of the surface area that is visible within v from the vantage point of the

sensor, the angular separation between adjacent rays, the distance of the sensed surface from the sensor origin, and the voxel space resolution. An estimate of the visible surface area that intersects v can be determined from an estimate of the surface's local curvature. Using a first order model, a plane equation can be calculated from the range data intersecting v [92], and the area of the surface's intersection with v can then be estimated by the area of the plane's intersection with v . More accurate estimates can be calculated using higher order derivatives of the surface curvature.

A more sophisticated error model would further associate with each range point p a probability density function that the space in the neighborhood of p contains surface data. Rather than incrementing a single voxel per point, this function could be used to weight a neighborhood of voxels per point during voxelation.

For simplicity and efficiency, and to avoid the sensitivities inherent in estimating surface derivatives from range data [60], we use a constant threshold value k , and change a voxel's value to **S** only if the number of range image points that fall within it exceeds k . A constant threshold is effective when ρ is much greater than the inter-point separation.

An example of the effect of image noise filtering is illustrated in Figure 4.3. A range image of an isolated object, a carved duck, lying on a flat table is shown from the sensor view in part **a**, and from a rotated view in part

b. This particular image was acquired with an early laboratory prototype of the autosynchronous scanning method [97] which had fallen into disuse, and subsequently exhibited significantly more noise than is typical in ranging systems. In particular, spurious noise is evident in a pyramidal pattern above the table, and at the edges of the scanning frustum.

The **S**-valued voxels of the unfiltered voxelated image (i.e. $k = 1$), with resolution $\rho = 10$ mm, are shown in part c. The results of filtering the voxelated image with a threshold value of $k=5$ is shown in part d. It can be seen that the filter has removed a majority of the obvious noise.

The effect of increasing the value of k for this image is observed in Graph 4.4, which plots the change in the number of **S**-valued voxels that are removed by the filter as a function of k . Let the filter for a voxelated range image $\mathbf{V}$ with threshold value k be denoted as $g_k(\mathbf{V})$, and let the size of the resultant filtered voxel space be denoted as $\|g_k(\mathbf{V})\|$. For each value of k , Graph 4.4 plots the change $\Delta\|g_k(\mathbf{V})\|$ in the size of the filtered voxel space ;

$$\Delta\|g_k(\mathbf{V})\| = \|g_{k-1}(\mathbf{V})\| - \|g_k(\mathbf{V})\| \quad (4.6)$$

Graph 4.4 indicates that the number of additional voxels removed with increasing k tends towards a constant as k increases, after the majority of the noise has been removed. A suitable value under these conditions would be $k=5$.

Figure 4.3: Rank Filter

Figure 4.4: Plot of filter threshold (k) vs. delta voxels removed ($\Delta\|g_k(\mathbf{V})\|$)

4.2 Pose Space Quantization

The pose space Θ of a model comprises all possible translations and orientations of the model. Given a continuous measurement space, there exist an infinite number of such poses. The quantization of $\mathcal{R}^3$ into a discrete voxel space, however, imposes an equivalent quantization of pose space, which results in a finite-sized discrete pose space.

4.2.1 Voxel Occupancy

The equivalence of the quantization of $\mathcal{R}^3$ and Θ can be described in terms of a model's *voxel occupancy*. Denoted as $\{v \cap M_{\vec{\theta}}\}$, the voxel occupancy is the specific set of voxels that intersect with the surface of a model pose $M_{\vec{\theta}}$. As we consider model surfaces only, the voxels interior to a closed model are not included in the voxel occupancy set.

For a given $M_{\vec{\theta}}$ the value of $\vec{\theta}$ can be perturbed by a small amount $\Delta\vec{\theta}$ in any direction without changing the voxel occupancy :

$$\exists \Delta\vec{\theta} \text{ s.t. } \{v \cap M_{\vec{\theta}}\} = \{v \cap M_{\vec{\theta} + \Delta\vec{\theta}}\} \quad (4.7)$$

The only time when this will not hold is when a surface of $M_{\vec{\theta}}$ intersects exactly with the border of two or more voxels, in which case a pose change

in certain directions, no matter how small, may add or remove some voxels from the voxel occupancy set.

The $\Delta\vec{\theta}$ which surrounds each $\vec{\theta}$ form a neighborhood of pose values that share the same voxel occupancy. Each such neighborhood can be considered a distinct coordinate in a discrete pose space. The size and shape of each discrete pose space coordinate is a non-linear function of the model geometry and the voxel space resolution and offset. Rather than calculate the exact dimensions of each pose coordinate individually, we approximate the discrete pose space under the assumption that the coordinates are distributed uniformly throughout the space. The approximation method handles the translation and rotation dimensions separately, and while described for the most general six dimensional pose space, can be scaled down to any subspace.

The size of the translational dimensions are determined directly from the voxel space resolution ρ . Any point on the model's surface can be translated by at most $\pm\rho$ before it falls within another voxel. The extent of any translational dimension of a discrete pose coordinate is therefore equal to ρ .

The extent of the discrete pose coordinate rotational dimensions are limited by a solid angle ε . Rotating the model in any direction by ε would translate any point on the surface of $\mathbf{M}$ by at most ρ . The value of ε is a function of ρ and the model radius $r_{\mathbf{M}}$, which is the furthest distance from

Figure 4.5: Geometric Construction Relating Model Radius ($r_{\mathbf{M}}$), Voxel Space Resolution (ρ), and Angle Resolution (ε)

the centroid of $\mathbf{M}$ to any point on it's surface. The value of ε is calculated from the geometric construction of Figure 4.5 as :

$$\varepsilon = 2 \sin^{-1} \left(\frac{\rho}{2r_{\mathbf{M}}} \right) \quad (4.8)$$

4.2.2 Viewing Sphere

In Section 3.1, a pose was defined as a rigid transformation between the sensor coordinate reference frame and an object-centered coordinate reference frame, the origin of which was arbitrarily placed at the centroid of the object. Each model pose therefore positions the model within the sensor's reference frame.

A dual interpretation of the pose describes the sensor location with re-

spect to the object reference frame, and gives rise to the notion of a *viewing sphere*. The viewing sphere is a sphere centered at the object frame origin, with a radius no less than the model radius r_M . Inverting the rotational dimensions of the pose positions the sensor frame at some point on the surface of the sphere. The sensor frame is oriented so that the sensor's field of view is always pointed toward the model frame origin, i.e. the sensor-frame z-axis is aligned radially with the viewing sphere.

The viewing sphere representation has been used in both Aspect Graph [98] and Extended Gaussian Image [99] methods. In these cases, features were extracted from the object independent of the rotation about the sensor frame z-axis, and so the spherical system was described in two dimensional polar coordinates. Unlike these cases, our method depends upon the z-axis rotation. An example of our viewing sphere is illustrated in Figure 4.6, where ϑ and φ represent the polar rotations about the object frame x - and y -axes, and ψ represents the additional z-axis rotation. The transformation between the sensor frame **B** and object frame **A** can be expressed as :

$$\mathbf{A} = \text{Rot}(x, \vartheta) \text{Rot}(y, \varphi) \text{Rot}(z, \psi) \mathbf{B} \quad (4.9)$$

Our objective is to generate a uniform solid angular sampling of the three dimensional spherical coordinate frame, so that each distinct pose coordinate is equidistant from its neighbors. A sampling which has uniform increments

Figure 4.6: Spherical Coordinate Frame

in ϑ and φ will result in a non-uniform solid angular sampling, with samples clustered at the sphere poles and dispersed at the equator.

A method of generating a near-uniform sampling of a sphere coordinate frame is to base the sampling upon an icosahedron. An icosahedron is a polyhedra with 20 equal sized triangular faces, and is the regular Platonic solid with the largest number of faces. A uniform sampling of the sphere can be generated by calculating the spherical coordinate at the center of each polygon face. The resulting 20 coordinates are typically too far separated to adequately represent the viewing sphere, and so additional coordinates are interpolated by connecting the midpoints of the triangular edges. Each trian-

gle is subdivided into 4 smaller triangles, and the new vertices are extended to intersect with the surface of the sphere. Each such interpolation results in a fourfold increase in the number of polygonal faces. The resulting sampling is not uniform, but is close enough to a uniform sampling for practical purposes. The subdivisions are often referred to as levels of the icosahedron, with 80 faces for level 1, 320 faces for level 2, etc. The first four icosahedron levels, and their associated number of faces and average angular sampling resolution ε , are illustrated in Figure 4.7.

Our usage of the viewing sphere is similar to that of the Aspect Graph methods, in that the sphere is used to generate a distinct rotation of the model for compilation in a database and comparison with an image. One difference is that the Aspect Graph methods extract features from the model which are rotationally invariant around the viewing direction, whereas in our case, the additional z -axis rotation is required.

Ideally, the icosahedron angular sampling increment will be just larger than the ε value calculated from equation 4.8. In some cases it may be practical to choose an icosahedron sphere with a somewhat larger sampling increment in order to reduce the size of the pose space. The z -axis rotation ψ is decoupled from the polar rotations, and is independently set to be the smaller of ε or the icosahedron sampling increment. The size of the three dimensional purely rotational pose spaces that result from setting the z -axis

a) level 0, faces=20, $\varepsilon=41.0$ deg

b) level 1, faces=80, $\varepsilon=19.9$ deg

c) level 2, faces=320, $\varepsilon=9.9$ deg

d) level 3, faces=1280, $\varepsilon=4.7$ deg

Figure 4.7: Icosahedron and Derived Subdivisions

<i>level</i>	<i>number of faces (n_f)</i>	ε	$n_f \times \lceil 360/\varepsilon \rceil$
0	20	41.0	180
1	80	19.9	1520
2	320	9.9	11840
3	1280	4.7	98560

Table 4.1: Icosahedra Pose Space Sizes

rotation equal to the icosahedron sampling increment are tabulated in Table 4.1.

Ideally, we would choose the icosahedron which has an angular increment ε closest to and just greater than the value determined from equation 4.8. In practice, due to the large jumps in cardinality between icosahedron levels, we often choose a icosahedron with an angular increment somewhat coarser than the ideal. This may cause ambiguities, the effects of which can be discounted through a number of means in both the class generation and tree traversal stages.

4.3 Summary of Chapter

In this chapter, quantization was presented as a practical technique to represent the otherwise infinite decision space. The decision space was quantized into a voxel space, which was described with a three parameter model. A

method of generating the voxel space from range image data was presented based upon the space carving principle. The two sampling effects of this method were described, and a discrete domain rank filter was described to remove image noise.

The concept of voxel occupancy was introduced, and it was discussed how the quantization of the decision space results in the effective quantization of the pose space. The viewing sphere was presented as an approximation of the discrete pose space, and the uniform angular resolution of the discrete pose space was related to the icosahedron level of the viewing sphere.

Chapter 5

Class Generation

Each class is composed of the set of points residing on the non-occluded surfaces of a model in a specific pose. There exists a separate class for every surface point of every model pose that intersects with the decision space origin. In the continuous domain, there are an infinite number of points on the surfaces of any model pose, and an infinite number of poses. Quantization of the decision space results in a finite number of discrete surface points for each model pose, and the quantization of pose space results in a finite number of model poses.

The straightforward method of generating the class set for a given model is to orient the model at each purely rotational coordinate of the discrete pose space, voxelate the rotated model, and apply a self-occlusion filter to the voxel space. We call the set of **S**-valued voxels resulting from each operation a *rotation-template* (*r-template*). This is the pattern of voxels that is observable

in the image by rotating the model, at the specified translation, to the pose value. A class is then equivalent to an r-template translated so that some member $\mathbf{S}$ -valued voxel lies at the decision space origin.

If the tessellation of the discrete pose space were exact, then generating the classes in the above described manner would be sufficient. Any possible continuous model pose imaged in the scene would result in a distinct voxel occupancy. The pose space tessellation is approximate rather than exact, primarily because it is desirable to reduce the number of classes by coarsening the discrete pose space resolution.

Let n_{Θ_R} be the size (i.e. number of distinct coordinates) of the purely rotational subspace Θ_R of the discrete pose space Θ . The value of n_{Θ_R} was established in Section 4.2 to be dependent upon the angular resolution of the viewing sphere, which was a function of the icosahedra level. For each model, the number of classes (n_ω) will equal the product of n_{Θ_R} and the average number of non-occluded $\mathbf{S}$ -valued voxels per r-template (n_s). As a conservative upper estimate, we set n_s to be half of the surface area of the model, normalized by the voxel space resolution ;

$$\begin{aligned} n_\omega &= n_{\Theta_R} n_s \\ &= n_{\Theta_R} \frac{\text{surface area of model}}{2\rho^2} \end{aligned} \tag{5.1}$$

<i>icosahedron level</i>	<i>surface area/ρ^2</i>		
	10^1	10^2	10^3
0	9e2	9e3	9e4
1	7.6e3	7.6e4	7.6e5
2	5.9e4	5.9e5	5.9e6
3	4.9e5	4.9e6	4.9e7

Table 5.1: Discrete Pose Space Sizes (n_w)

Some values of n_w for various icosahedra levels are tabulated in Table 5.1. For a given model radius r_M , our choice of voxel space resolution ρ leads to a discrete pose space ideally based upon a level 3 icosahedron (from Equation 4.8). For the region in which we tend to operate, which is $(\text{surface area})/\rho^2 > 10^2$, the number of classes is large for the level 3 icosahedron.

One way to reduce number of classes is through *translation thinning*, which reduces the number of purely translational classes representing each r-template. In the current definition, a distinct class is generated for each $\mathbf{S}$ -valued voxel of each r-template. For an r-template with n_s elements, there will be n_s separate classes which differ only in that they are offset from one another translationally.

The reason for this redundancy is to add robustness to the recognition process. If the data were perfect, then an image of the model pose represented by the above described r-template would result in n_s perfect matches. Noise

and occlusions can, however, cause any individual voxel to be absent in real image data. Having a number of classes describe each r-template increases the chances that at least one of them is present in the image data.

As recognition will occur if even only one translation is represented by the class set, it is not necessary to generate all possible translational classes. Translation thinning reduces the value of n_ω by randomly selecting a fraction $\mathcal{K}$ of the possible classes to represent each r-template, which linearly reduces the number of classes ;

$$n_\omega = \mathcal{K} n_{\Theta_R} n_s \quad (5.2)$$

5.1 Stable Voxels

We are motivated to further reduce n_ω by selecting a viewing sphere which is coarser than the ideal. This results in increased quantization errors. The effects of quantization errors can be reduced through the use of *stable voxels*, which are those voxels most likely to be **S**-valued, given a model pose close to the nominal class pose. The stable voxels are those **S**-valued voxels most resilient to quantization errors, and provide a more robust description of a model pose. As such, only the stable voxels are included in a class definition.

To calculate the stable voxels for each class, the model pose is randomly perturbed to a set of pose values close to the nominal class pose, and a voxel

space is generated at each pose. Let $\vec{\theta}_a$ be a discrete pose space coordinate which is a nominal class pose. The model is perturbed to a random pose value $\vec{\theta}_b = \vec{\theta}_a + \Delta\vec{\theta}$, where $\Delta\vec{\theta} = (\Delta x, \Delta y, \Delta z, \Delta\vartheta, \Delta\varphi, \Delta\psi)$. The magnitudes and directions of both the translation and rotation components of $\Delta\vec{\theta}$ are uniformly distributed and fall within the discrete pose space resolution;

$$\Delta x^2 + \Delta y^2 + \Delta z^2 < \rho^2/4 \quad (5.3)$$

$$\Delta\vartheta^2 + \Delta\varphi^2 + \Delta\psi^2 < \varepsilon^2/4 \quad (5.4)$$

These perturbed poses represent a sampling of that neighborhood of continuous pose values which is mapped to the nominal discrete pose value. The stable voxels are the **S**-valued voxels which are present in the majority of these perturbed model poses. They are the most likely to be present in an arbitrary continuous model pose in the neighborhood of the nominal pose.

The calculation of stable voxels once again uses a voxel space as a histogram. The pose of the model is repeatedly perturbed n_T times about the nominal class pose coordinate, and the values of each non-occluded **S**-valued voxel are incremented for each voxelated model pose perturbation. Typically, n_T is set to 100, which is considered to be a sufficient number of trials. The stable voxels are those voxels that are present in greater than a certain threshold percentage t_v of the n_T trials.

The stable voxels generated at various threshold values for a toy boat model are illustrated in Figure 5.1. The model is shown in part a, and a model pose in part b. In parts c to f, the stable voxels are displayed for various thresholds. In Graph 5.2, the ratio of the number of stable to total S-valued voxels is plotted for this model pose as a function of threshold t_v .

Removing the unstable voxels from the class voxel occupancy set is one way of improving robustness to quantization errors, but cannot be relied upon to handle all quantization errors. It is evident from Graph 5.2 that the number of stable voxels tends to zero as the value of t_v tends to 1. This indicates that the resolution of the pose space quantization is slightly coarser than the correct value for the given voxel space resolution, which is a desirable criterion to reduce the number of classes to a manageable size.

5.2 Matching Criterion

The use of stable voxels was motivated by the requirement to reduce the number of classes by coarsening the discrete pose space resolution. The stable voxels represent the nominal model pose over a larger than ideal range of poses for the given voxel space resolution.

The objective of resolution coarsening was to reduce the number of classes and ultimately improve efficiency. This efficiency enhancement comes at the

Figure 5.1: Stable Voxels at Various Thresholds

Figure 5.2: Ratio of Stable Voxels to Total Voxels (n_s/n_v) vs. Threshold (t_v)

cost of reliability. In order to address the reliability of the use of stable voxels, we consider the criterion used to evaluate matching a class.

The degree of match of a class with an image is measured with a correlation function $r(,)$. Let $\mathbf{V}_a$ be a voxel space resulting from voxelating the image $\mathbf{I}$. Let the translated r -template corresponding to $M_{\vec{\theta}_b}$ be represented as a voxel space $\mathbf{V}_b$. We assume that the two voxel spaces have the same bounds, offset, and resolution.

Let the function $\delta(v,k)$ identify whether or not voxel v has value k :

$$\delta(v, k) = \begin{cases} 1 & \text{if value of } v = k \\ 0 & \text{otherwise} \end{cases} \quad (5.5)$$

A correlation function which measures the similarity of the distribution of $\mathbf{S}$ -valued voxels in the two spaces is then :

$$r(\mathbf{V}_a, \mathbf{V}_b) = \frac{\sum_i \sum_j \sum_k \delta(v_a(i, j, k), \mathbf{S}) \delta(v_b(i, j, k), \mathbf{S})}{\sum_i \sum_j \sum_k \delta(v_a(i, j, k), \mathbf{S})}, \quad v_a \in \mathbf{V}_a, v_b \in \mathbf{V}_b \quad (5.6)$$

where the summations occur over the full extent of the i, j , and k indices within the voxel spaces. This correlation function is equivalent to the *absolute sum of difference* metric, which is a standard in template matching [100].

5.2.1 Quantization Errors

If the granularity of the quantization of the discrete pose space were sufficiently fine so that each continuous model pose in the scene resulted in the exact voxel occupancy of some class, then the matching criterion of equation 5.6 would sufficiently discriminate correct classes by evaluating to value 1. Errors occurring in both image voxelation and class generation can, however, cause the patterns to match less than exactly, even when the actual model pose is very close to a nominal class pose.

The effect of quantization errors for similar pose values is illustrated in the following example. A model with radius $r_{\mathbf{M}} = 91.7$ mm at pose $\vec{\theta}_{\mathbf{a}}$ was voxelated in $\mathbf{V}_a$ with resolution $\rho = 10$ mm. At this resolution, from equation

4.8 the exact angular resolution for the discrete pose space was $\varepsilon = 6.25$ degrees, which lies between the angular resolutions of the level 2 and level 3 icosahedra. In generating $\mathbf{V}_a$, only the stable voxels were included, with stable voxel threshold $t_v = 0.75$.

The pose was next translated to a random value $\vec{\theta}_b = \vec{\theta}_a + \Delta\vec{\theta}$, where $\Delta\vec{\theta} = (\Delta x, \Delta y, \Delta z, 0, 0, 0)$. The magnitudes of the translational components of $\Delta\vec{\theta}$ were within the pose space resolution, i.e. equation 5.3 held for $\rho = 10$ mm. The new model pose $M_{\vec{\theta}_b}$ was voxelated into voxel space $\mathbf{V}_b$, and the two voxel patterns were compared by evaluating $r(\mathbf{V}_a, \mathbf{V}_b)$. Note that the stable voxel filter was not used in generating $\mathbf{V}_b$, so that all S-valued voxels resulting from $\vec{\theta}_b$ were included.

This procedure was repeated 1000 times with $\vec{\theta}_a$ randomly selected from the set of viewing sphere coordinates. The results were plotted in Graph 5.3.

This experiment was then repeated for purely rotational perturbations, i.e. $\vec{\theta}_b = \vec{\theta}_a + \Delta\vec{\theta}$, $\Delta\vec{\theta} = (0, 0, 0, \Delta\vartheta, \Delta\varphi, \Delta\psi)$. The data set and the least squares regression parabola were plotted in Graph 5.4. Once again, the stable voxel threshold $t_v = 0.75$ was used to generate $\mathbf{V}_a$, and the magnitude of the rotational components of $\Delta\vec{\theta}$ were within the pose space resolution of the level 2 icosahedron, i.e. equation 5.4 held for $\varepsilon = 9.9$ degrees.

The conclusion drawn was that even when the magnitude of the per-

Figure 5.3: Translational Pose Perturbation vs. Correlation ($t_v = 0.75$)

turbations were well within the discrete pose space resolution, mismatches signified by a correlation value $r(V_a, V_b) < 1$ occurred due to quantization errors.

The previous experiment (with purely rotational perturbations) was repeated for stable voxel threshold values of $t_v = 0.50$, and 0.25 , and the resulting data and least squares parabolas were plotted in Graphs 5.5 and 5.6. The least squares parabolas for the three trials were plotted in Graph 5.7. As expected, an increase in t_v tended to improve the correlation measure.

Figure 5.4: Rotational Pose Perturbation vs. Correlation ($t_v = 0.75$)

5.3 Matching Threshold

Both quantization errors and image noise can cause the correlation function to evaluate to less than unity for a correct class comparison. We wish to determine whether there exists a threshold value r_t of the correlation function $r(\cdot)$ which can be used to discriminate between correct and incorrect matches.

A value for r_t can be determined by generating and comparing sets of correct and incorrect matches. The boat model of Figure 5.1 was transformed to a set of 1000 discrete pose space coordinates $\{\vec{\theta}_{a_i}\}, i = 1..1000$. For each $\vec{\theta}_{a_i}$, another model pose $\vec{\theta}_{b_i}$ was randomly generated to fall within the discrete

Figure 5.5: Rotational Pose Perturbation vs. Correlation ($t_v = 0.50$)

pose space coordinate of $\vec{\theta}_{a_i}$. The magnitudes and directions of the $\vec{\theta}_{b_i}$ were uniformly distributed and equations 5.3 and 5.4 held.

The model was transformed to each corresponding $\vec{\theta}_{a_i}$ and $\vec{\theta}_{b_i}$, and the respective voxel spaces V_{a_i} and V_{b_i} were generated. In generating the V_{a_i} , a stable voxel threshold of $t_v = 0.75$ was used. The values of $r(V_{a_i}, V_{b_i})$ were evaluated and stored. This data set was referred to as the *within* data, as each perturbed pose $\vec{\theta}_{b_i}$ was within the discrete pose space resolution of it's corresponding pose $\vec{\theta}_{a_i}$. This set therefore represented samples of the correlation function values for a correct match.

Figure 5.6: Rotational Pose Perturbation vs. Correlation ($t_v = 0.25$)

Figure 5.7: Regression of Rotational Pose Perturbation vs. Correlation

This procedure was repeated for a set of poses $\{\vec{\theta}_{a_j}\}, j = 1..1000$, with the values of $\vec{\theta}_{b_j}$ representing discrete pose coordinates different from the $\vec{\theta}_{a_j}$. Each $\vec{\theta}_{b_j}$ was a random perturbation from a discrete pose coordinate distinct from and not neighboring $\vec{\theta}_{a_i}$. This set of values $r(\mathbf{V}_{a_j}, \mathbf{V}_{b_j})$ therefore represented a sample of the correlation function values resulting from a mismatch. This set of values and was stored as the *between* data.

Both data sets were plotted in Graph 5.8. In order to relate the correlation value as a combined function of both translation and rotation perturbations, the x -axis of the graph was in normalized units ;

$$x = \frac{(\Delta x^2 + \Delta y^2 + \Delta z^2)^{1/2}}{\rho} \quad (5.7)$$

$$+ \frac{(\Delta \vartheta^2 + \Delta \varphi^2 + \Delta \psi^2)^{1/2}}{\varepsilon} \quad (5.8)$$

Within a class, when equations 5.3 and 5.4 hold, the magnitude of the normalized perturbation value did not exceed 1, which accounted for the clustering of the *within* data in the $x \in [0, 1]$ region.

The least squares regression lines and 99 % confidence intervals of the two data sets within the $x \in [0, 1]$ region were plotted in Graph 5.9. The separation of the confidence intervals suggested that the correlation threshold value of $r_t \approx 0.3$ could be used to accurately discriminate between correct and

Figure 5.8: Comparison of Pose Perturbation vs. Correlation

incorrect classes. In practice, similarities based upon scene clutter motivates us to increase this threshold value, as described in Chapter 8.

5.4 Summary of Chapter

This chapter described how classes are generated. It was shown that the number of classes for a given model could be quite large if the viewing sphere icosahedron level were chosen based upon the ideal angular resolution value for a given decision space quantization resolution.

Two methods were presented to reduce the size of the class set, ulti-

Figure 5.9: Confidence Intervals

mately increasing the efficiency of the recognition process. The first was translation thinning, which linearly reduced the size of the class set by randomly generating only a percentage of the translationally offset classes for a given r -template. The second was to base the viewing sphere on a coarser than ideal icosahedron level.

Coarsening the discrete pose space could lead to quantization errors which could reduce the reliability of the method. It was described how quantization errors could be reduced through the use of stable voxels, which are those voxels most resilient to quantization errors.

The matching criterion was defined as an absolute sum of differences cor-

relation function, and a set of experiments examined the behavior of the correlation function for comparing and scoring model poses which were separated within the resolution. It was observed that the value of the correlation function reduced as the separation of the model poses increased. An experiment demonstrated that increasing the stable voxel threshold increased the bias of the correlation function.

An experiment established confidence that a threshold correlation value could be used to discriminate between correct and incorrect matches.

Chapter 6

Tree Construction

6.1 Background on Trees and Tree Construction

The tree structure is among the most well utilized representations in object recognition and software systems. Indeed, the tree structure is so fundamental for structuring information that there is evidence of its use in antiquity [101, p. 41],[85, p. 405]. If we consider the leaf nodes as representing the items of a database, then the main advantage of tree structuring is its ability to access an item in logarithmic time.

The objective of decision tree construction is to compose a decision tree from a set of classes which is well-suited for the given task at hand. Suitability can be evaluated by one or more measures, such as *node count*, *leaf profile*, *min/max/expected traversal cost*, and *misclassification rate*. Moret

[82] points out that even for a moderate number of features and classes the number of possible decision trees is huge, which rules out the feasibility of simply generating and comparing the suitability of all alternatives. A method is therefore desired which optimizes a set of metrics during tree construction.

Moret has also shown that it is not possible to simultaneously optimize all metrics. He defines a set of metrics to be *compatible* if a tree exists which simultaneously optimizes the set. It was shown that the worst case traversal cost, expected traversal cost, and node count are all pairwise incompatible metrics [82].

The *expected traversal cost* is the most commonly optimized metric in decision tree construction. Let a tree T have n_L leaf nodes and n_I internal nodes. Let each internal node t have an associated cost $C(t)$, which is the expense of executing the evaluation function for that node. Each node t will also have an associated probability $p(t)$, which is the prior probability of the class (if t is a leaf node) or the feature (if t is an internal node).

Let π_j denote the *path* to node t_j , which is defined as the unique sequence of n_{π_j} nodes from the root of T to leaf node t_j ;

$$\pi_j = \{t_1, t_2, \dots, t_j\} \quad (6.1)$$

The *cost* $C(\pi_j)$ of traversing π_j is defined as the sum of the costs of every internal node in π_j ;

$$C(\pi_j) = \sum_{k=1}^{n_{\pi_j}-1} C(t_k) \quad (6.2)$$

Note that the leaf node does not contribute to the cost of traversal, as it has no decision function, and therefore no associated cost value.

Similarly, let the *weighted cost* $\mathcal{W}(\pi_j)$ be defined as the sum of the cost of every internal node in the path, weighted by the node probabilities ;

$$\mathcal{W}(\pi_j) = \sum_{k=1}^{n_{\pi_j}-1} C(t_k)p(t_k) \quad (6.3)$$

The expected traversal cost E_T of the tree T is then the sum of the weighted path costs for each leaf node ;

$$E_T = \sum_{j=1}^{n_L} \mathcal{W}(\pi_j) \quad (6.4)$$

The expression for E_T assumes a depth-first non-tentative traversal policy, which, when applicable, is the most efficient of all traversal policies. For other policies, such as breadth-first or tentative (backtracking), this expression is not valid. It seems that there has been no effort directed towards optimization methods for alternate traversal policies, and we assume that

optimizing the tree structure with respect to equation 6.4 will likely result in a good (i.e. near optimal) structure for other traversal policies.

Reliability metrics, such as *misclassification rate*, are also good candidates for optimization in tree design. We do not optimize for reliability here for two reasons. First, such an optimization generally requires a knowledge of the reliabilities of the features and/or classes, which we do not assume to know. Second, Moret has shown that the misclassification rate and expected traversal costs are pairwise incompatible, meaning that they cannot be simultaneously optimized in a single tree design. We will address the tradeoffs between reliability and efficiency empirically in Chapter 8.

6.2 Optimal Methods

A tree structure is said to be *balanced* if all leaf nodes are of approximately equidistant path length from the root node. At one time decision trees were constructed based upon the belief that a balanced tree structure minimized E_T [101, p 45]. This was proven incorrect by Garey [102] who showed that a balanced tree structure could result in an expected testing cost arbitrarily greater than the optimal.

6.2.1 Dynamic Programming

One way to generate an optimal tree is through Dynamic Programming. The method generally applies to problems which adhere to Bellman's *principle of optimality* [103, p.83], wherein an optimal solution can be constructed from optimal subsolutions. For example, if we assume a tree structure T with root node t_0 to be minimal with respect to E_T , then the subtrees T_j rooted at every internal child node t_j of t_0 must also each minimize E_{T_j} . In addition to the expected testing cost, the principle of optimality applies to other tree metrics, such as worst case testing cost.

Dynamic Programming is a bottom-up method which first constructs subsolutions, and then merges them into a global solution. Initially, all leaf nodes are merged into sets where a single common feature value is unresolved. These sets are then merged to form larger sets of an increasing number of unresolved features. At each merge the relevant metrics are calculated and stored.

The merging continues until a single set is generated which has all features unresolved, representing the root node of the tree. The resulting merged sets form a rooted directed acyclic graph. Each node of the graph represents a merged set which points to the sets from which it was merged. A tree which is optimal with respect to one of the propagated metrics can then be generated

from a simple non-tentative depth-first traversal of the graph, following the path with the optimal cost.

For an n -dimensional decision space of k -ary features, the time complexity of Dynamic Programming is $O(nk(k+1)^{n-1})$, which makes the method unsuitable for moderately large values of n .

6.2.2 Branch-and-Bound

Branch-and-Bound is a top down method for optimal tree construction. The method is applicable to problems where, for any partial solution, a conservative bounding estimate exists for the optimal value of the measure to be optimized. At each step, the Branch-and-Bound method expands the partial solution with the best conservative estimate, jumping (backtracking) to alternate branches as warranted.

In cases where the objective is to minimize a metric, such as worst-case traversal cost, the conservative estimate used is an upper bound. Alternately, when the metric is to be maximized, the conservative estimate used is a lower bound. For example, assume we wish to generate a tree with minimum worst-case traversal cost. An upper bound estimate of this value for the entire tree would simply be the sum of all variable evaluation costs, as any traversal could be no more expensive than the cost of evaluating every variable. Similarly, if a partial solution exists, then an upper bound estimate

for any leaf of the partial solution would be the sum of the cost of the path to that leaf and the cost of evaluating all remaining variables.

Moret [84] has defined the *activity* of a variable as the product of the variable's testing cost and the prior probability that the variable will be tested. He has determined that activity is the only estimate which can be suitably used as an upper bound for expected traversal cost.

An advantage of the Branch-and-Bound method over Dynamic Programming is that it can be used to optimize measures that do not satisfy Bellman's principle of optimality, such as the diagram storage cost. The main disadvantage is its exponential time complexity. In the worst case, if the conservative estimate is weak, the method can result in an exhaustive enumeration of all possibilities.

6.2.3 Huffman's Algorithm

While not general, Huffman's Algorithm is noteworthy for its simplicity. The method was discovered first by Huffman and then independently by Zimmerman [85, p.406], and deals with the special case where all evaluation costs are uniform, and there are weights associated only with the leaf nodes. When dealing with decision trees, this corresponds to the case where the class prior probabilities are known but feature priors are unknown, and all variable evaluation costs are equal.

Let a *weighted path* be the path length multiplied by the weight of the leaf node to which the path resolves. Huffman's Algorithm can be used to generate binary trees with minimal *weighted external path length*, which is the sum of weighted paths to each leaf node in the tree. When the weights represent prior probabilities, the method minimizes expected traversal cost.

The method works from the bottom up, initially selecting the pair of leaf nodes with the smallest weights. These two nodes are merged into a single node with the sum of the two as its weight. The procedure is applied recursively to the new node set, terminating when only one node remains. The minimal tree solution is then expanded by substituting all merged nodes by their component pairs.

Unfortunately, Huffman's Algorithm is not applicable for general decision tree construction where both feature probabilities and non-uniform evaluation costs are important. Also, if at any iteration more than one leaf node pair has minimal weights, then the method arbitrarily chooses between them. In the case where all leaf node priors are identical, which is true in our problem, the method is therefore not applicable.

6.3 Sub-Optimal Methods

Any method which generates an optimal tree T with minimal E_T is known to be NP-complete [101, p.45] so it is no surprise that both Dynamic Programming and Branch-and-Bound methods are of exponential time complexity. For a sufficiently large decision space, the accepted practical approach has been to generate sub-optimal solutions based on heuristics [82, p.609]. The main advantage of sub-optimal heuristic methods is that they are efficient, ideally executing in polynomial time. The disadvantage is that the solution is not truly minimal with respect to E_T . It is worthwhile to note that true optimization is often a subjective design objective: in many cases, a sub-optimal solution can serve effectively.

Sub-optimal tree construction proceeds as a top-down process. Initially the complete set of classes is associated with the root node and is partitioned into subsets. A new child node is then branched off of the root node for each class subset. The process repeats for each new node until the node class sets are small enough to terminate at leaf nodes.

The node class sets are partitioned at each step by selecting a discriminating feature (or set of features) using a heuristic criteria, of which there are two main alternatives. The *balance* heuristic (also known as the *splitting* heuristic) attempts to spawn equal-sized subtrees from each internal node.

It does so by choosing that feature which divides the node class sets into roughly equal sized sets. The *information* heuristic (also known as the *entropy* heuristic) chooses that feature which will, upon interrogation, yield the greatest amount of added information to the process.

6.4 Implemented Method

In our particular problem the decision space is large, which makes optimal tree construction infeasible. Rather than restricting ourselves to one particular sub-optimal method, our approach has been to generate and compare the results of a number of heuristic criteria. All of the implemented methods were based upon a common top-down recursive procedure, described in pseudocode in Figure 6.1. Using this framework, the `select_feature()` function was implemented using four alternative heuristic feature selection criteria.

6.4.1 Heuristic Feature Selection Criteria

6.4.1.1 Balance Heuristic

The first feature selection method was based on the balance heuristic. For each node, a feature was selected that was present in roughly half of the classes in the node class set. The selected feature therefore partitioned the node class set into two subsets of roughly equal size. If more than one feature

```

node *build_tree(class_set  $\Omega$ )
{
    if (size( $\Omega$ ) < min_size)
    {
        node->type = leaf ;
        node->class_set =  $\Omega$  ;
    }
    else
    {
        v = select_feature( $\Omega$ ) ;
        partition_class_set( $\Omega$ , v,  $\Omega_T$ ,  $\Omega_F$ ) ;

        node->type = intermediate ;
        node->feature = v ;
        node->true_child = build_tree( $\Omega_T$ ) ;
        node->>false_child = build_tree( $\Omega_F$ ) ;
    }

    return(node) ;
}

```

Figure 6.1: Tree Construction Pseudocode

provided an equally good partition, then one was chosen randomly between the alternatives.

6.4.1.2 Max Information Heuristic

The second implementation was based on the information heuristic. For a given class set, that feature was selected which was most common to the member classes. This heuristic was labeled the *Max* Information Heuristic because at each node it selected the feature which was common to the maximum number of classes in the node class set.

6.4.1.3 Min Information Heuristic

The Min Information Heuristic feature selection criterion was the dual of the Max Information Heuristic. The feature was selected that was common to the smallest non-empty subset of the node class set.

6.4.1.4 Random Heuristic

The fourth and final method was to simply randomly select a feature, without any consideration of its characteristics. This criterion produced trees with arbitrarily good or bad, but likely average properties. The randomly generated trees were used as a basis of comparison of the three preceding methods.

6.4.2 Comparison of Feature Selection Criteria

The most meaningful comparison of the four criteria is an evaluation of the efficiency and reliability of tree traversal using real data under typical conditions. Such a comparison is presented in chapter 8. Here, we generate a set of trees using the various criteria and compare their metric properties.

Trees were generated for the duck and boat objects. Both objects could be positioned within a four dimensional table-top pose space, which allowed 2 dimensions of translation along the plane of the table, 1 rotational dimension around the table normal, and 1 translational dimension of the camera with respect to the table along the table normal.

A sample tree was constructed for each object using the Balance, Max Information, and Min Information feature selection heuristics. Three trees were constructed for each object using the Random criterion. The metric properties of the trees were compiled in Tables 6.1 and 6.2.

The results indicate that the Balance Heuristic produces trees with the most attractive metric properties, having the least node count and the smallest E_T and maximum (*worst-case*) traversal cost. While the minimum (*best case*) traversal cost was largest for the Balance Heuristic trees, this is rarely an important metric.

The tree with the second best metric properties resulted from the Max

<i>heuristic</i>	<i>node count</i>			<i>traversal cost</i>		
	<i>total</i>	<i>empty</i>	<i>non-empty</i>	<i>min</i>	<i>max</i>	E_T
		<i>leaves</i>	<i>leaves</i>			
balance	647	0	324	8	19	9.7
max info	1277	246	393	2	24	13.5
min info	1125	0	563	2	563	283
random 1	5885	2540	403	2	195	68.6
random 2	5893	2534	413	3	262	77.9
random 3	5359	2268	412	3	225	75.1

Table 6.1: Duck Tree Metrics

<i>heuristic</i>	<i>node count</i>			<i>traversal cost</i>		
	<i>total</i>	<i>empty</i>	<i>non-empty</i>	<i>min</i>	<i>max</i>	E_T
		<i>leaves</i>	<i>leaves</i>			
balance	1247	0	624	9	13	10.4
max info	3703	1067	785	2	46	18.2
min info	2185	0	1093	2	1093	548
random 1	7921	3173	788	3	136	43.2
random 2	9045	3730	793	2	197	56.0
random 3	8087	3271	773	2	139	51.8

Table 6.2: Boat Tree Metrics

Information Heuristic. These trees had roughly twice the number of nodes as the balanced trees, and only a slightly larger E_T and maximum traversal cost.

The metric properties of the trees resulting from the various feature selection criteria can be appreciated by considering the sequence with which space is interrogated during tree traversal. Under the Max Information Heuristic, as each node is visited, that point of space is interrogated which is common to the greatest number of remaining classes. Under the assumption that all classes have equal priors and are therefore equally likely to occur, the Max Information Heuristic tree traversal will thus interrogate the point which is most likely to intersect with a surface of the object, if the initial hypothesis H_0 is true. The traversal will therefore direct the sampling to points which are most likely to fall on the surface of the object, which is an intuitively appealing strategy. If H_0 is false, and the initial seed point did not fall on the object's surface, then each step of the tree traversal will interrogate a point of space with no special propensity to intersect with a surface, and will likely resolve to an empty leaf node.

The main structural difference between the Max Information and Balance trees is that the former results in a large number of leaf nodes that contain empty class sets. In a balanced tree, there are no paths that lead to empty class sets, resulting in fewer overall nodes, and accordingly better metric

properties. Traversing the Balanced tree will always resolve to a leaf node class set which is non-empty. The initial hypothesis H_0 will therefore always be supported, subject to further validation.

The Min Information Heuristic results in trees which sample those points in space which are least likely to fall on the object's surface. This is an intuitively unappealing strategy, and indeed results in trees with poor metric properties. These trees have structures which, among the alternatives, are the closest to being linear, with by far the largest E_T and maximum traversal cost. Indeed, the trees produced by this criterion had consistently poorer metric properties than any of the randomly generated trees. This suggests that the Min Information Heuristic is a poor feature selection criterion, and conversely, further supports the belief that its dual, the Max Information Heuristic, may well be a good criterion. It will be seen in the experimental result of Chapter 8, that, despite the metric properties, there are cases when the tree constructed with the Min Information Heuristic outperform all other trees in efficiency.

6.4.3 Complexity

In constructing a tree, the `build_tree()` routine of Figure 6.1 is called recursively for each new node. In any tree, the number of internal nodes n_I is one less than the number of leaf nodes n_L . If k classes are encoded per leaf

node, then given n_ω classes, the number of leaf nodes for a balanced tree is ;

$$n_L \approx n_\omega/k \quad (6.5)$$

We conclude, then, that the total number of nodes $n_T = O(n_\omega)$ and that `build_tree()` is executed $O(n_\omega)$ times.

The two main procedures within `build_tree()` are `partition_class_set()` and `select_feature()`. In its simplest form, `partition_class_set()` determines the class membership of a particular voxel by looping through each voxel of each class. If we assume that on average a class contains n_s voxels, then `partition_class_set()` requires $n_s \times n_\omega$ operations.

The `select_feature()` routine executes similarly for the four distinct heuristics. In both information heuristics, for each call to `select_feature()` a voxel histogram is constructed. Each voxel value in the histogram indicates the number of classes containing that voxel, and the feature is selected by identifying the maximum peak of the histogram. Histogram construction entails visiting each class in turn and incrementing the histogram values of member voxels, requiring $n_s \times n_\omega$ operations for the entire class set. Peak determination is an $O(n_s)$ operation.

Rather than constructing the histogram for each class set every time `build_tree()` is invoked, it is more efficient to construct the histogram at the offset from the initial complete class set, and maintain it by removing

and adding classes accordingly prior to calling `build_tree()` for a new class set. At any iteration it is therefore only necessary to loop through an often small portion of the classes to determine voxel membership. This reduces the computational expense but not change the complexity expression, as the reduction is in the form of a multiplicative factor.

The implementation of `select_feature()` for the Balance Heuristic is similar to that for the two information heuristics, and the complexity expressions are the same for both. The difference is that the average rather than the peak value of the histogram is identified in the case of the Balance Heuristic.

In the Random Heuristic, a feature is arbitrarily selected. To be meaningful, it is important that the selected features are members of at least some classes in the class set. We therefore maintain the histogram as a convenient representation for determining class membership of any randomly selected feature.

The complete time complexity expression for constructing a tree is $O(n_s n_\omega^2 + n_s n_\omega)$. Given the large number of classes, this can be a time consuming operation. Due to the recursive nature of the tree construction process, the straightforward implementation can also be memory consuming.

6.4.4 Inclusion Lists

The time expense of the `build_tree()` routine was reduced by partitioning the problem into two steps. In the first step, a set of intermediate representations called *inclusion lists* ($\mathcal{L}$) was calculated and stored. In the second step, the $\mathcal{L}$ set was used to generate the trees as described above. The time saving resulted primarily from parallelizing the first step, which comprised a large number of independent processes.

Each inclusion list $\mathcal{L}_j$ is a list of classes that contain a unique voxel v_j . Each voxel element of the histogram used in tree construction contains the number of classes which intersect that voxel. The inclusion list for each voxel also contains the identity of the intersecting classes. There exists one $\mathcal{L}_j$ for each histogram voxel, and they are precalculated and stored during the first processing step. In the second processing step, the $\mathcal{L}_j$ are accessed directly by voxel index.

The benefit of precalculating and storing the inclusion lists is that the `partition_class_set()` routine is reduced to a simple table lookup, replacing one of the $O(n_s n_\omega)$ operations within `build_tree()`. The number of operations for the various steps and the resultant complexity expression of tree construction with and without the use of inclusion lists are tabulated in Table 6.3. While the use of inclusion lists did not reduce the complexity

<i>routine</i>	<i>implementation</i>	
	<i>without $\mathcal{L}$</i>	<i>with $\mathcal{L}$</i>
<code>partition_class_set()</code>	$n_s n_\omega$	1
<code>maintain_histogram()</code>	$n_s n_\omega$	$n_s n_\omega$
<code>select_feature()</code>	n_s	n_s
<code>build_tree()</code>	$n_\omega(n_s n_\omega + n_s n_\omega + n_s)$ $= 2n_s n_\omega^2 + n_s n_\omega$	$n_\omega(1 + n_s n_\omega + n_s)$ $= n_s n_\omega^2 + n_s n_\omega + n_\omega$
<i>complexity</i>	$O(n_s n_\omega^2 + n_s n_\omega)$	$O(n_s n_\omega^2 + n_s n_\omega + n_\omega)$

Table 6.3: Tree Construction Complexity

expression, it did reduce the multiplicative factor.

For each voxel v_j , the inclusion list $\mathcal{L}_j$ is calculated by looping through each class ω_k to see if v_j is a member of ω_k . This requires $O(n_s n_\omega)$ operations. To generate the entire set of inclusion lists, this is repeated for each of the n_s voxels, resulting in a complexity of $O(n_s^2 n_\omega)$. The computation of each $\mathcal{L}_j$ is independent, and therefore amenable to parallel implementation. For m parallel processor resources, the time expense can be reduced to $\frac{1}{m}O(n_s^2 n_\omega)$.

As an example, the inclusion lists were generated using a parallel computer with a varying number of processor resources. The parallel computer was a network of workstations organized as a heterogeneous parallel computer using the interprocess communication software PVM. The tests were executed for the duck object with a four dimensional pose space, which resulted in the generation of approximately 8,000 inclusion lists. The actual

Figure 6.2: # of Processors (m) vs. Inclusion List Generation Time (t)

time results, plotted in Figure 6.2, indicated that increasing the number of processors indeed did indeed reduce the time required to generate the inclusion lists. While the actual time saved was not large in this particular example, when the dimension of the model pose space is larger, this improvement factor would lead to significant time savings. For example, to generate the inclusion lists for the boat model with the full six dimensional pose space required over two days of actual processing time using a network of 18 parallel processor resources.

The time expense of generating trees with and without the use of inclusion lists was compared in Table 6.4. Trees were generated for the duck

<i>heuristic</i>	<i>without $\mathcal{L}$</i>	<i>with $\mathcal{L}$</i>
balance	26.6	9.0
max info	39.5	17.0

Table 6.4: Tree Generation Time (seconds)

object with a four dimensional pose space using the Max Information and the Balance Heuristics. While the actual time expense was reduced when the inclusion lists were used, the combined cost of calculating the inclusion lists and then generating an individual tree was greater than simply generating the tree without the use of inclusion lists. This was true even when the inclusion lists were generated using the parallel platform.

An additional expense of generating the $\mathcal{L}$ set explicitly is incurred because not all generated $\mathcal{L}$ are used in constructing the tree. For example, in the case of the duck object with a four dimensional pose space, 2232 inclusion lists were generated, but only 218, or were used to generate the balanced tree. For the boat with a 6 dimensional pose space, the size of the $\mathcal{L}$ set was 24290, but only 1814 (7.5%) were used to generate the balanced tree. Of course, it is not possible to know in advance which inclusion lists will be required to generate a tree under a specific heuristic selection criterion. An alternative to generating the entire $\mathcal{L}$ set would be to cache only those calculated during tree generation, for use in subsequent tree generation calls. There is

however an advantage to reducing the actual computation time required to construct a tree, which can be large. For example, the time to generate a tree for the duck model with a six dimensional pose space using the Maximum Information Heuristic was approximately 25 hours, using inclusion lists. An extrapolation from Table 6.4 indicates that, without the use of precalculated inclusion lists, the tree generation time would be 2 to 3 times greater.

6.5 Summary of Chapter

In this chapter, we defined some metric properties of trees, and reviewed decision tree construction methods. Some of the construction methods were optimal with respect to the metrics, but all optimal methods were NP-complete and therefore computationally expensive. We therefore concluded that sub-optimal methods were preferred.

A recursive implementation of sub-optimal tree construction was presented. Using this as a framework, four tree construction methods based upon four distinct heuristic feature selection criteria were developed. Trees were generated using the various heuristic criteria and their metric properties were compared. It was concluded that the Balance and Max Information criteria resulted in the trees with the most attractive metric properties.

The computational complexity of the tree construction method was de-

terminated. It was described how the tree construction method could be made less time consuming by partitioning it into two steps, and parallelizing the first step. This was implemented across a network of heterogeneous processors, and timing test were presented which demonstrated an improvement in time performance.

Chapter 7

Tree Traversal

Tree traversal is the process of visiting a subset of the tree nodes in an ordered fashion. If the objective of a particular traversal is to terminate at a leaf node, as in our case, then the simplest type of traversal is *non-tentative depth-first*. Starting from the root node, the sequence of nodes visited passes from parent to child, and terminates when a leaf node is encountered. This is the most efficient way of traversing from the root to a leaf node, requiring only $\log n_L$ visits for a tree with n_L leaf nodes.

If the image data were perfect, then the non-tentative depth-first traversal would be guaranteed to always produce the best solution, i.e. that template which correlates best for the given seed. In practice, however, errors caused by noise, occlusions, and quantization effects cause some voxels in the image voxel map to have a value different from the ideal. The result is that any branching decision in the non-tentative depth-first traversal may be incorrect,

causing the traversal to find a less than ideal or incorrect solution.

The *reliability* of a tree traversal method is a measure of its ability to resolve to a correct, or near correct, solution, given that errors exist in the data. The reliability of tree traversal can be improved, and in some cases optimized, through the use of *backtracking*. Backtracking, or *tentative* traversal, allows the exploration of partial paths without necessarily committing to them. It recognizes that a path which may seem promising at one depth of exploration may prove less so as it is followed. If a particular path being followed becomes less attractive than another previously encountered partial path, the tentative traversal will switch attention back to the other more promising path. The reliability improvement which can be realized through the use of backtracking comes at the expense of efficiency, as a larger portion of the tree nodes are visited during traversal.

A traversal which is optimal with respect to reliability is said to be *guaranteed*. In our case, a guaranteed traversal will always resolve to the leaf node with the best correlation. Traversals which are not guaranteed are said to be *sub-optimal* or *heuristic*. If a guaranteed traversal exists, it places a lower bound on the efficiency of reasonable heuristic methods, as it makes no sense to use a sub-optimal method which is less efficient than guaranteed traversal. Indeed, if a guaranteed traversal exists, the only reason to consider the use of heuristic methods is to improve efficiency, if the subsequent

reliability degradation can be tolerated.

7.1 Guaranteed Traversal

Guaranteed traversal is based upon a branch-and-bound search. In general, branch-and-bound (or A*) methods are known to be optimal if, at each visited node, there exists a conservative bounding estimate of the quantity under optimization. In our case, the correlation value is maximized, and the bounding estimate at node t_j is an upper estimate of the correlation of any leaf template descending from t_j .

We denote the bounding estimate at t_j as $\hat{r}_j^+$, and determine its value as follows. The tree is first preprocessed so that each t_j encodes the size $n_{\pi_j}^+$ of the largest template associated with any of the leaf nodes descending from the subtree with t_j as its root. If t_j is itself a leaf node, then $n_{\pi_j}^+$ is trivially the size of its largest associated template.

As the tree is traversed, a running count $n_{\pi_j}^S$ is maintained along each partially expanded path π_j of the number of S-valued voxels encountered along π_j . A count $n_{\pi_j}^T$ is also maintained which indicates the number of nodes along π_j that are descended from the TRUE branch of their parent node. For example, if all nodes along a π_j have descended from TRUE branches, then $n_{\pi_j}^T$ would be equal to the length of π_j . If a single node descended from a

FALSE branch, then $n_{\pi_j}^T$ would be one less than the path length, etc.

An expression for the conservative estimate $\hat{r}_j^+$ of the maximum correlation value of any template rooted at t_j is :

$$\hat{r}_j^+ = \frac{n_{\pi_j}^+ + n_{\pi_j}^S - n_{\pi_j}^T}{n_{\pi_j}^+} \quad (7.1)$$

The estimate $\hat{r}_j^+$ is the ratio of the number of possible **S**-valued voxels remaining along π_j to the ideal number. The numerator component $n_{\pi_j}^S - n_{\pi_j}^T$ is the difference between the possible and the actual number of **S**-valued voxels encountered while traversing π_j . It is therefore a count of the number of missed **S**-valued voxels along the path, which will necessarily reduce the correlation value of any descendant templates.

If all subsequent template elements were present in the data, then the measured correlation value would equal the estimate. Any further differences would reduce the measured correlation value, which makes $\hat{r}_j^+$ an upper estimate.

During Guaranteed traversal, as each node t_j is encountered, $\hat{r}_j^+$ is calculated and compared to two values. The first is a threshold r_t which is the minimal acceptable correlation for a correct solution. The second is the current maximum correlation value r^+ for any leaf node which has been visited during the traversal. At any visited t_j , if $\hat{r}_j^+$ exceeds both r_t and r^+ ,

then the current path potentially leads to the best solution, and is expanded further. Otherwise, it is certain that any matches at leaf nodes descendant from t_j will necessarily correlate less than either r_t or r_i^+ so π_j is pruned from further evaluation. Whenever a leaf node is encountered, a full correlation is evaluated for each of its associated templates, and the value of r^+ is updated.

Whenever a leaf node is encountered or a path is pruned, traversal continues by backtracking to the immediate parent node and evaluating the alternative path. This continues until all remaining paths have been either ruled out or expanded fully to the leaf node. The traversal is guaranteed to identify the solution with maximum correlation value, and at no additional cost returns all encountered solutions that exceed r_t .

The limitation of the Guaranteed traversal is that it is computationally expensive. In the worst case every node is visited, making it no better than (and when overhead is considered, somewhat worse than) a linear pass through an unordered list of the n templates. In the best case, it can perform closer to $O(\log n)$. One can imagine artificially constructing trees and scenes so that the traversal operates in either the best or worst case manner. How efficiently it will perform in any given situation depends on both the tree structure and the scene data.

One way to improve efficiency would be to increase the value of r_t , which would result in the earlier pruning of a greater number of subtrees. To demon-

Figure 7.1: Traversal Correlation Threshold (r_t) vs. Efficiency (E_1), Guaranteed

strate this effect, the Guaranteed traversal efficiency was measured and plotted for a variety of r_t values. The efficiency measure used was the ratio of the number of template correlations required for an exhaustive brute-force template matching, vs. the average number of correlations actually calculated per seed voxel during Guaranteed traversal. This is the E_1 metric defined in Section 8.2.1. A balanced tree was used, constructed from the boat model under 4 dof conditions. The tree encoded 180 templates. The experiment was repeated for 4 images of the boat object in different configurations, and the results are plotted in Graph 7.1.

The results indicate that the efficiency improved as the value of r_t increased. The best efficiency, realized when $r_t = 1.0$, was approximately 5, which is much less than $\log n$. As the r_t value decreased, the worst efficiency quickly degraded to that of exhaustive correlation. Under realistic conditions, a suitable r_t value is likely to be no greater than 0.75. *We therefore conclude that, under these conditions, Guaranteed traversal is not efficient enough to be practical.*

The Guaranteed traversal method returns the maximum correlating template across all image seeds. By varying the procedure slightly, it is also possible to return the best match for each seed, or all matches which exceed r_t . Both of these variations are less efficient than the described procedure, and not necessarily more useful in our context.

7.2 Heuristic Traversal Methods

With the aim of improving efficiency, four heuristic traversal methods were implemented. Within each method, at each visited t_j the value of $\hat{r}_j^+$ is calculated. If ever $\hat{r}_j^+$ falls below r_t , then the current path is discarded from further evaluation.

7.2.1 Best-First

The Best-First traversal heuristically attempts to find the template with maximum correlation value by immediately following the most promising path. At each visited t_j , the $\hat{r}^+$ is calculated for each child of t_j , and that branch is followed which maximizes this value. If a tie exists, then the TRUE child is followed.

This is a simple non-tentative depth-first traversal, with a complexity of $O(\log n_L)$ for n_L leaf nodes.

7.2.2 Beam

The Beam traversal employs a limited form of backtracking. Rather than making the decision to follow a branch based just on the direct descendants of each node, the Beam traversal looks ahead into the tree to a greater depth. At each visited node t_j , all paths of depth d are expanded, and the $\hat{r}^+$ calculated. The direct descendant of t_j whose depth $d - 1$ subtree contains the node with the largest $\hat{r}^+$ value is followed. Whenever a leaf node is encountered, the correlation value of each of its associated templates is calculated. Any encountered templates whose values exceed r_t are added as a solution.

The decision of which child to follow is based upon more information than

the Best-First case, so the Beam traversal tends to be more reliable. It is also less efficient than Best-First traversal : for each path node, all subtrees of depth d are expanded, resulting in a complexity of $O(2^d \log n_L)$.

7.2.3 N-Error

The N-Error traversal was motivated by error coding methods in information theory. A path is considered as a string of the binary symbols TRUE and FALSE, indicating which branch was followed from the parent nodes. Any symbol along the path can be incorrect, and we assume that each symbol has the same probability of being in error.

Initially a Best-First traversal is invoked, which results in a path π from the root to some leaf node. It is then assumed that any branching decision could have been in error. For each node in π , the traversal then backtracks and takes the alternate branch, which generates a new path resolving at a different leaf node.

We can assume that there are e erroneous symbols in any path, and expand all alternatives. Given average path length of $\log n_L$ for n_L leaf nodes, π will be on average length $\log n_L$, and the number of possible variations from π with e is :

$$P \left(\begin{matrix} \log n_L \\ e \end{matrix} \right) = \frac{\log n_L!}{(\log n_L - e)!} \quad (7.2)$$

The complexity of this traversal is therefore $O(\log^e n_L)$.

7.2.4 Heuristic Branch-and-Bound

The Heuristic Branch-and-Bound traversal combines aspects of both the N-Error and the Guaranteed traversals. It is based on the assumption that any branching errors encountered along a traversed path will be randomly distributed along that path in a uniform way.

Under this assumption, for any partial path π_j of length n_{π_j} , we can estimate the correlation of a template associated with a descendant leaf node. This estimate is denoted $\hat{r}_j$, and is the ratio of the number of encountered to ideal **S**-valued voxels :

$$\hat{r}_j = n_{\pi_j}^S / n_{\pi_j}^T \quad (7.3)$$

Unlike $\hat{r}_j^+$, $\hat{r}_j$ may fall on either side of the resultant measured correlation value. It therefore cannot be used to drive an optimal traversal. In many cases, however, $\hat{r}_j$ will provide a closer estimate than $\hat{r}_j^+$, particularly when the partial path length is still small.

The traversal follows the same branch-and-bound regime as Guaranteed traversal, pruning all branches for which $\hat{r}$ is less than either r_t or $\hat{r}_t^+$. The value of r_t should be set a little lower than in previous searches to accommodate the possibility that $\hat{r}$ now can be an underestimate. Also, pruning

Figure 7.2: Traversal Correlation Threshold (r_t) vs. Efficiency (E_1), Guaranteed and Heuristic Branch-and-Bound

along a particular path should start only after the value of $n_{\pi_j}^T$ is sufficiently large that the value of $n_{\pi_j}^S$ is based upon a significant sample.

Although this traversal has the same $O(n)$ complexity as Guaranteed traversal, it will on average perform more efficiently as it uses a more liberal pruning criterion. The cases where the reliability of this traversal will suffer is when there exist a series of erroneous voxel values which are encountered early in an otherwise error-free path (which leads to a correct solution). Under the stated assumption that branching errors are uniformly distributed along a path, this is an unlikely occurrence.

The advantage of Heuristic Branch-and-Bound traversal is that it is both more reliable than the other heuristic traversals and more efficient than Guaranteed traversal. Graph 7.2 plots the efficiency of both the Guaranteed and Heuristic Branch-and-Bound traversals for variety of r_t values. The image was of the boat model, and the tree encoded 180 templates of the boat with a 4 dof pose space. At the highest value of r_t , the two methods have the same efficiency. As the r_t value is reduced, while the Guaranteed traversal efficiency quickly dropped off to unity, the efficiency of the Heuristic Branch-and-Bound traversal degraded much more gracefully. Even at quite low r_t values (under 0.60), the efficiency increase was still relatively close to that of the maximum, at $r_t = 1.0$.

7.3 Probe Comparison

Figure 7.3 illustrates the probed voxels for the various traversal methods. The same balanced tree and seed voxel was used for each traversal. The Beam traversal had a branching depth $d = 2$, and the N-Error traversal allowed $e = 1$ errors.

The seed voxels are rendered in blue, the S-valued voxels in green, and the non-S-valued voxels in red. As expected, the more expensive traversals resulted in more extensive probing of the neighborhood around the seed.

(a) Best-First (6 probes)

(b) Beam (64 probes)

(c) N-Error (7 probes)

(d) Branch-and-Bound (88 probes)

Figure 7.3: Comparison of Probes for Different Traversals

<i>method</i>	<i>complexity</i>
Best-First	$\log n_L$
Beam	$2^d \log n_L$
N-Error	$\log^e n_L$
H Branch-n-Bound	n_L
Guaranteed	n_L

Table 7.1: Tree Traversal Complexity Expressions

7.4 Summary

Four methods of tree traversal were presented, and their computational complexities analyzed. For convenience, the complexity expressions were tabulated in Table 7.1. One of these traversals was optimal, guaranteed to always return the highest correlating solution.

Experiments revealed the dependence of traversal efficiency E_1 and the correlation threshold r_t for the Guaranteed and Heuristic Branch-and-Bound traversals. It was concluded that the Guaranteed Branch-and-Bound traversal was not efficient enough to be practical.

Chapter 8

Experimental Results

The described method was implemented and tested on a number of objects under a variety of scene conditions. The primary purpose of the implementation and tests was to demonstrate the effectiveness and identify the performance of the method. Secondary objectives were to empirically determine suitable values of the threshold parameter r_t , and to determine if there existed significant performance differences for various combinations of tree and traversal types.

8.1 Experimental Setup

8.1.1 Objects

The tests were performed on two objects, a toy boat, and a carved duck. A characteristic common to each object is that they were roughly the same

size, which allowed us to quantize at a single resolution.

Models were constructed for each object by first acquiring a covering set of range images at a number of different aspects (~ 10). These range images were then combined into a single registered point cloud, from which a triangular mesh polyhedron (*t-mesh*) was generated.

8.1.2 Trees

The tree representations were constructed directly from each t-mesh. Trees were constructed of each object for a 4 dof and 6 dof pose space. The 4 dof pose space assumed that there existed some restriction of the orientation of the object with respect to the sensor. An example of such a scenario would be a table top sensing environment, where the object is restricted to lie in a stable set of orientations. The object could be rotated around the sensor azimuth (z-axis), or translated in 3 dimensions. The 6 dof pose space represented the most general case, wherein the object could be positioned at any orientation or translation. In order to reduce the time to generate the tree representations, the space of object orientations was restricted so that only the top half of the object was considered. Two rotational dimensions of the six dimensional pose space were therefore only half-dimensions.

For the 4 dof tests, trees were constructed using all heuristic methods, resulting in *Balance*, *Max-Info*, *Min-Info*, and three *Random* trees for each

object. In the 6 dof tests, only the *Balance*, *Max-Info*, and *Min-Info* trees were constructed to reduce preprocessing.

8.1.3 Images

Range images were acquired with two different sensors; an autosynchronous scanner [97], and a Biris sensor. The Biris sensor was mounted on a linear scanning platform, and produced an image of parallel profiles. The autosynchronous scanner generated cylindrical profile images.

Images were acquired of each object under a variety of isolated and cluttered scene conditions. In the isolated scenes, only the object of interest was present, except possibly for a planar background. In the cluttered scenes, there were a number of objects arbitrarily placed in the image. There were also some examples of occluded scenes, where the object of interest was either partially covered by some other object, or partially outside of the sensor's field of view.

Twenty images were acquired for each object and each pose space in each scene condition. The object was randomly positioned within its pose space for each image, and an attempt was made to position the object approximately uniformly throughout its pose space for each image set.

8.2 Performance Metrics

8.2.1 Efficiency Metrics

Let a tree T containing n nodes encode t templates. Let a traversal of T for all s image seeds result in c correlation calculations. The efficiency E_1 of a given traversal is defined as the ratio of the number of correlation calculations required for exhaustive template matching with that number actually executed during traversal ;

$$E_1 = \frac{t \times s}{c} \quad (8.1)$$

8.2.2 Reliability Metrics

The reliability of the method is a measure of its ability to recognize correctly. We measure reliability in two ways, the first of which is by manually inspecting the solutions. A match was judged to be correct if it was close enough to the object's pose that an ICP would drive it to a true solution. We denote this metric as R_1 . R_1 is binary-valued. For a single recognition event, R_1 can be either 1 (indicating success) or 0 (indicating failure);

$$R_1 = \begin{cases} 1 & \text{if success} \\ 0 & \text{if failure} \end{cases} \quad (8.2)$$

The second reliability metric is denoted R_2 and was determined auto-

matically by examining the maximum correlation value r^+ of all solutions. If r^+ were greater than some threshold r_t then the recognition was deemed successful, and R_2 was set to r^+ . Otherwise, the recognition event had failed, and R_2 was set to 0.

$$R_2 = \begin{cases} r^+ & \text{if } r^+ \geq r_t \\ 0 & \text{if } r^+ < r_t \end{cases} \quad (8.3)$$

Of the two metrics, R_1 is the most faithful way to determine reliability. In an operational scenario, however, we desire a way to automatically determine recognition events, and so we rely more upon R_2 .

The effectiveness of R_2 depends largely on the value of r_t . There are two types of classification errors which can affect the fidelity of R_2 . A *false negative* occurs when a true match is rejected, as may be the result of excessive noise or occlusion. A *false positive* occurs when an incorrect match is accepted, such as when the scene content is close to that of the object's surface geometry.

In general, a tradeoff exists between the rates of false negative (α) and false positive (β), and our case is no exception. If the value of r_t is too high, then the chance of rejecting a correct solution is greater, and the false negative rate α will increase. A large r_t will also reduce the chance of accepting a poor solution so the false positive rate β will simultaneously decrease. Conversely, if r_t is too low, then α will decrease, and β will increase.

In the case of the Guaranteed traversal the solution will always have the highest possible value of r^+ , but will not necessarily be a correct match. For example, when the object is not in the scene, the highest correlating solution does not constitute a true recognition event. Also, in occluded scenes or noisy images, a correct match may have a lower correlation value than an incorrect match. In cases where the best solution is not guaranteed, the matches may nevertheless be close enough to be correct. In many non-trivial cases the highest correlating solution returned from a heuristic search does indeed constitute a correct match.

8.3 Boat Object Tests

8.3.1 Correlation Threshold Determination

An experiment was designed to determine whether or not a value of r_t exists which could be used to differentiate between true matches and false positives. Using the boat object with a 4 dof pose space, a set of 20 images were acquired under isolated conditions. This image set represented a sample of the population of images where a true positive match was possible, and was called the *positive* image set. The resolution of quantization was set to $\rho = 0.015\ m$, which was roughly 10% of the maximum radius of the object. A best-first traversal of the balanced tree was applied to each image in the

	<i>Image Set</i>	
	<i>positive</i>	<i>negative</i>
<i>set size</i>	20	22
<i>min R₂</i>	0.738	0.75
$\bar{R}_2$	0.841	0.826
<i>max R₂</i>	0.922	0.894
<i>s</i>	0.061	0.039
<i>R.R.</i>	< 0.8231	> 0.8373

Table 8.1: R_2 Statistics ($\rho = 0.015\ m$)

set with the r_t value set to zero so that every traversal generated at least one solution. The R_2 values resulting from each traversal were recorded, and the average $\bar{R}_2$ and standard deviation s of these values were calculated.

This process was then repeated using a set of 22 images of arbitrary cluttered scenes which did not contain the boat object. Again, the best-first traversal of the balanced tree was applied with a threshold correlation value of $r_t = 0$. This image set was called the *negative* image set as it represented a population in which only negative or false positive values were possible. The statistics from both positive and negative trials were tabulated in Table 8.1.

A statistical t-test was applied to determine whether or not the population means differed significantly. The details of this calculation were compiled in Appendix A.1. The test results indicated that there was no evidence to

support the hypothesis that the values of $\bar{R}_2$ between the populations differed at a significance level of $\alpha=0.10$.

To reinforce this conclusion, the lower boundary of the 90% confidence interval, formulated as the rejection region (R.R.) of a t-test, was calculated for the positive trial. Similarly, the upper boundary was calculated for the negative trial. The details of these calculations were presented in Appendix A.5, and the results are tabulated in the last row of Table 8.1.

It was calculated that the rejection regions for the 2 trials did indeed overlap. The implication of this result was that, under conditions similar to those in the test, we are unable to simultaneously identify true recognition events and guard against false positives solely by examining the resultant correlation values of solutions. Selecting a threshold which was within the 90% confidence interval for the true matches would admit too many false matches, thereby increasing the false positive error rate to be $\beta \geq 0.10$. Similarly, for the false positive error rate to be controlled under $\beta < 0.10$, the threshold would intrude into the 90% confidence interval of the true matches, and the false negative error rate would be $\alpha \geq 0.10$.

One way to improve the discrimination between the two populations was to increase the fineness of the quantization resolution. The above experiment was repeated for a quantization resolution of $\rho = 0.010$ m. The details of the statistical test were compiled in Appendix A.2 and the results from the

	<i>Image Set</i>	
	<i>positive</i>	<i>negative</i>
<i>set size</i>	20	22
<i>min R₂</i>	0.655	0.622
$\bar{R}_2$	0.792	0.759
<i>max R₂</i>	0.875	0.872
<i>s</i>	0.060	0.068
<i>R.R.</i>	< 0.7735	> 0.7697

Table 8.2: R_2 Statistics ($\rho = 0.010m$)

trials were tabulated in Table 8.2. In this case the two population means were found to differ at a significance level of $\alpha = 0.10$. The 90% rejection regions did not overlap, as detailed in Appendix A.6. A suitable threshold to discriminate between the true and false positive matches could be $R_2 = 0.770$.

The above process could be used in general to empirically determine a suitable value for r_t , particularly in cases where sample images are available which are very representative of those to be encountered in an operational scenario.

8.3.2 4 DOF Isolated Scenes

An experiment was designed to determine the relative performance of various combinations of trees and traversal methods. In particular, it was desired to know if certain combinations of trees and traversals were significantly more

efficient and/or reliable than others.

The experiment involved a set of 10 test images of the boat object under a 4 dof pose space in isolated scene conditions. The boat was placed arbitrarily in different parts of the scene and the sensor height was varied with respect to the table top to a number of locations. The recognition method was then applied to each image for each combination of tree and traversal. In total, 6 trees (Balance, Max-Info, Min-Info, Random-1, Random-2, Random-3) and 4 traversals (Best-First, N-Error, Heuristic Branch-and-Bound, Beam) were tested, resulting in 24 trials. For each traversal, the correlation threshold value was set to $r_t = 0.75$, and the quantization resolution was $\rho = 0.015$.

A summary of the results were compiled in Tables 8.3 and 8.4, and an Analysis of Variance (ANOVA) was performed for both the R_2 and E_1 metrics. An example of a successful trial was illustrated in Figure 8.1. In this and all subsequent figures of recognition results, a number of metrics from the particular trial illustrated are listed in parenthesis above the figure caption. In particular, the tree traversal/design type (e.g. BF/r2), the E_1 metric, the time t taken for recognition, the maximum correlation value r^+ , and the number of successful solutions returned. Note that all tests were executed on a single MIPS R8000 processor, rated at ≈ 7.5 MFLOPS. The code was non-optimized C++. The timing results indicate only the elapsed time to traverse the tree and calculate the correlation value for the resulting solu-

tions. The total elapsed time was larger by a small factor. It is believed that the times reported are a realistic lower bound on the time to execute a suitably optimized implementation of the method on the given hardware platform.

	<i>tree</i>					
<i>traversal</i>	<i>Bal</i>	<i>Max-Info</i>	<i>Min-Info</i>	<i>Rand-1</i>	<i>Rand-2</i>	<i>Rand-3</i>
Best-First	.777	.672	.786	.508	.421	.347
Beam	.867	.865	.427	.428	.509	.495
N-Error	.854	.851	.795	.787	.863	.797
Branch-n-Bound	.873	.867	.795	.873	.873	.873

Table 8.3: $\bar{R}_2$ Data Summary

	<i>tree</i>					
<i>traversal</i>	<i>Bal</i>	<i>Max-Info</i>	<i>Min-Info</i>	<i>Rand-1</i>	<i>Rand-2</i>	<i>Rand-3</i>
Best-First	95.0	170.5	177.6	391.2	588.5	6058.6
Beam	7.9	5.5	45.0	35.4	33.9	22.8
N-Error	17.8	16.9	2.4	11.2	9.2	9.0
Branch-n-Bound	4.2	4.0	5.9	8.4	8.8	7.7

Table 8.4: $\bar{E}_1$ Data Summary

8.3.2.1 R_2 ANOVA

To determine whether any particular tree/traversal combinations were more reliable, Fisher's Protected Least Significant Difference (LSD) test was applied to the data. The detailed calculations were compiled in Appendix A.3.1.

(BF/r2, $E_1 = 307$, $t = 0.21$ sec, $r^+ = 0.93$, n sols = 7)

Figure 8.1: Boat, 4 dof, Isolated Scene

First, a standard ANOVA was applied and it was concluded that, at significance level $\alpha = 0.10$, there existed a difference between the average R_2 values of the various trials. Following the ANOVA, Fisher's LSD was calculated as $LSD = 0.257$.

Based upon the LSD value, the 24 trials were partitioned into two equivalence sets, as illustrated in Table 8.5. Set B were those combinations which were less reliable, and comprised all random trees with the Best-First and Beam traversals. In addition, the Min-Info tree with the Beam traversal was a member of set B. The remainder of the tree/traversal combinations were more reliable and were included in set A.

The conclusions drawn from this result were :

1. The least reliable combinations were the randomly generated trees with either the Best-First or the Beam traversal. Intuitively this is an attractive result, as the Best-First and Beam traversals were the least extensive traversals, and the random trees are the most likely to have arbitrary performance.
2. The Min-Info tree with the Beam traversal was also among the less reliable combinations
3. For the N-Error and Heuristic Branch-and-Bound traversals, the choice of tree had a less significant effect on reliability.

4. For the Balance and Max-Info trees, the choice of traversal method had a less significant effect on reliability.

<i>traversal</i>	<i>tree</i>					
	<i>Bal</i>	<i>Max-Info</i>	<i>Min-Info</i>	<i>Rand-1</i>	<i>Rand-2</i>	<i>Rand-3</i>
Best-First	A	A	A	B	B	B
Beam	A	A	B	B	B	B
N-Error	A	A	A	A	A	A
Branch-n-Bound	A	A	A	A	A	A

Table 8.5: R_2 LSD Partition

8.3.2.2 E_1 ANOVA

The above procedure was repeated for the efficiency metric E_1 . The results were summarized in Table 8.4 and the details of the calculations were compiled in Appendix A.4.

The result from the ANOVA was that there was a difference between the efficiency of the trials at significance level $\alpha = 0.10$. An application of Fisher's LSD showed that the only tree/traversal pair which differed significantly was the combination of the Random-3 tree with the Best-First traversal, which had a much higher efficiency than any other combination. This result was believed to be anomalous: the tree was randomly generated, and so its properties could be extreme. Indeed, the reliability results from Table 8.3

<i>traversal</i>	<i>tree</i>					
	<i>Bal</i>	<i>Max-Info</i>	<i>Min-Info</i>	<i>Rand-1</i>	<i>Rand-2</i>	<i>Rand-3</i>
Best-First	Y,X	X,W	W	V	U	-
Beam	Z	Z	Z,Y	Z,Y	Z,Y	Z
N-Error	Z	Z	Z	Z	Z	Z
Branch-n-Bound	Z	Z	Z	Z	Z	Z

Table 8.6: E_1 LSD Partition

indicated that this combination had the lowest $\bar{R}_2$ of all trials.

The Random-3 tree with Best-First traversal trial was removed from the data set, and Fisher's LSD was recalculated for the remaining 23 trials. Based upon this calculation the trials were partitioned into 6 equivalence sets, as illustrated in Table 8.6. The sets are labeled in lexicographic order by decreasing efficiency. The largest set (Z) comprised the least efficient combinations and contained every trial which did not use a Best-First traversal. Among the Best-First traversal trials, all of the random tree trials were the most efficient and differed significantly from each other. There was no difference between the Min-Info and Max-Info trees, and the Balance tree could be grouped in a set with some of the less efficient traversals.

The conclusions from this test were :

1. The combination of the Random-3 tree with the Best-First traversal produced anomalous results. This indicated that this randomly gener-

ated tree structure was not well suited for Best-First traversal. Curiously, the Random-3 tree was partitioned into the least efficient set (Z) for all other traversal methods.

2. The Best-First traversal was significantly more efficient than the other traversal methods for all tree designs.
3. The Beam, N-Error, and Heuristic Branch-and-Bound methods were the least efficient for all tree structures.

Figure 8.2 plots the combined results of the tests for both the $\bar{E}_1$ and $\bar{R}_2$ metrics. The reliability/efficiency tradeoff was evident. The data for the Best-First traversal for the various trees were labeled. The Balance, Max-Info, and Min-Info trees with the Best-First traversal and represented combinations which the tests indicated were both significantly more reliable and more efficient than the others. It is interesting that the Balanced tree was neither the most efficient nor the most reliable. Indeed, the Min-Info tree had both a higher reliability and a higher efficiency than the Balanced tree.

The $\bar{R}_1$ metric was calculated for each trial by manually inspecting the results. Figure 8.3 plots both the $\bar{R}_1$ and $\bar{R}_2$ metrics against efficiency $\bar{E}_1$ for all combinations of traversal method and tree design. The plot indicates that the $\bar{R}_1$ and $\bar{R}_2$ values are closely related, with $\bar{R}_2$ biased slightly negative

key: BF=Best-First, bal=Balance, max=Max-Info,
min=Min-Info, r1=Random1, r2=Random2

Figure 8.2: Efficiency ($\bar{E}_1$) vs. Reliability ($\bar{R}_2$), (boat/4dof/isolated)

from $\bar{R}_1$ by a nearly constant factor. This further supports the use of R_2 for automatically determining positive matches.

8.3.3 4 DOF Cluttered Scenes

Each of the 24 combinations of traversal method and tree design were tested against a set of 20 images of the boat object under cluttered scene conditions.

Figure 8.3: Efficiency ($\bar{E}_1$) vs. Reliability ($\bar{R}_1, \bar{R}_2$), (boat/4dof/isolated)

An example of a successful result was illustrated in Figure 8.4, and the $\bar{R}_1$ and $\bar{R}_2$ against $\bar{E}_1$ metrics were plotted in Figures 8.5. The numerical results were tabulated in Tables B.1, B.2, and B.3 of Appendix B.

As was the case under isolated scene conditions, the Best-First traversal combined with the Random-3 tree was both less reliable and much more efficient than the other combinations. The results of all trials were replotted for the lower range of $\bar{E}_1 < 600$ in Figure 8.6. The order of the combinations was roughly the same as in the isolated scene case, with the Best-First traversal proving to be the most efficient than the others. The Min-Info and Max-Info

(a) image

(b) solution

(c) image, rotated view

(d) solution, rotated view

(BF/max, $E_1 = 161$, $t = 0.65$ sec, $r^+ = 0.98$, n sols = 21)

Figure 8.4: Boat, 4 dof, Cluttered Scene

Figure 8.6: Efficiency ($\bar{E}_1$) vs. Reliability ($\bar{R}_1, \bar{R}_2$), (boat/4dof/cluttered)

8.3.4 6 DOF Isolated Scenes

The performance results from trials of the boat object with a 6 dof pose space for 20 images under isolated scene conditions were plotted in Figure 8.8 and the numerical results tabulated in Tables B.1, B.2, and B.3 of Appendix B.3. An example of a successful trial was illustrated in Figure 8.7. The Balance, Max-Information, and Min-Information trees were tested in combination with all 4 traversal methods. The correlation threshold was set to $r_t = 0.55$, which was lower than in the 4 dof case, and therefore more susceptible to false positives.

(a) image

(b) solution

(c) image, rotated view

(d) solution, rotated view

(BF/min, $E_1 = 28601$, $t = 2.2$ sec, $r^+ = 0.67$, n sols = 2)

Figure 8.7: Boat, 6 dof, Isolated Scene

Figure 8.8: Efficiency ($\bar{E}_1$) vs. Reliability ($\bar{R}_1, \bar{R}_2$), (boat/6dof/isolated)

The plot indicates that the reliability was lower and the efficiency higher than the 4 dof case: all three trials using the Best-First traversal had $\bar{E}_1 > 10^3$. The Best-First traversal combined with the Min-Information tree design was particularly outstanding, with an efficiency of $\bar{E}_1 = 25828$ and a respectable reliability of $\bar{R}_1 = .60$. Also notable was the Beam traversal with the Min-Information tree, which had a large efficiency ($\bar{E}_1 = 9504$), but zero reliability.

The lower $\bar{E}_1$ range of these data was also of interest and was replotted in Figure 8.9. Four combinations, the Beam and N-Error traversals with

the Max-Information and Balance tree designs, had $\bar{E}_1 > 10^2$. The Beam traversal with the Max-Information tree lies in a good region, with $\bar{E}_1 = 164$ and $\bar{R}_1 = 0.70$. The Heuristic Branch-and-Bound traversal combined with the Min-Information tree design was also notable, as it had a moderate efficiency of $\bar{E}_1 = 56$, but a perfect reliability.

Figure 8.9: Efficiency ($\bar{E}_1$) vs. Reliability ($\bar{R}_1, \bar{R}_2$), (boat/6dof/isolated)

8.3.5 6 DOF Cluttered Scenes

The arrangements of traversal/tree combinations in plots Figure 8.11 and Figure 8.12 of the 6 dof cluttered scene tests closely follow those of the isolated scene data. An example of a successful trial is illustrated in Figure 8.10. Once again, the Best-First traversal with the Min-Info tree design showed a huge efficiency of $E_1 = 28800$, although the reliability at $\bar{R}_1 = .05$ was lower scene tests in the isolated scene tests. Conversely, the other combinations using the Best-First traversal had reliabilities which were higher than in the isolated scene tests.

In the lower range plot (Figure 8.12), the cluster of 4 combinations within the $100 < \bar{E}_1 < 500$ interval remain the same as in the isolated case, although their ordering has changed somewhat. Also, in general the reliability has increased somewhat within this range.

(a) image

(b) solution

(c) image, rotated view

(d) solution, rotated view

(BF/max, $E_1 = 4135$, $t = 4.56$ sec, $r^+ = 0.58$, n sols = 1)

Figure 8.10: Boat, 6 dof, Cluttered Scene

Figure 8.11: Efficiency ($\bar{E}_1$) vs. Reliability ($\bar{R}_1, \bar{R}_2$), (boat/6dof/cluttered)

Figure 8.12: Efficiency ($\bar{E}_1$) vs. Reliability ($\bar{R}_1, \bar{R}_2$), (boat/6dof/cluttered)

8.4 Duck Object Tests

The recognition trials were repeated for a carved duck object with 4 and 6 dof pose spaces under isolated and cluttered scene conditions. Sets of twenty images were used for the 4 and 6 dof isolated scenes, and the 4 dof cluttered scene. The image set used for the 6 dof cluttered scene contained 21 images. The voxel resolution was set at $\rho = 0.10\text{m}$ for all trials.

8.4.1 4 DOF Isolated Scenes

The performance results from the tests of the duck object with a 4 dof pose space under isolated scene conditions were plotted in Figure 8.14 and the numerical results were tabulated in Tables B.13, B.14, and B.15 of Appendix B.5. An example of a successful trial was illustrated in Figure 8.13.

(a) image

(b) solution

(c) image, rotated view

(d) solution, rotated view

(BF/min, $E_1 = 224$, $t = 1.1$ sec, $r^+ = 0.90$, n sols = 66)

Figure 8.13: Duck, 4 dof, Isolated Scene

Figure 8.14: Efficiency ($\bar{E}_1$) vs. Reliability ($\bar{R}_1, \bar{R}_2$), (duck/4 DOF/isolated)

Figure 8.15: Efficiency ($\bar{E}_1$) vs. Reliability ($\bar{R}_1, \bar{R}_2$), (duck/4 DOF/isolated)

Two of the random tree designs combined with the Best-First traversal were very efficient ($\bar{E}_1 = 2679,916$) and moderately reliable ($\bar{R}_1 = 0.4, 0.6$ respectively). The lower range of these data were replotted in Graph 8.15. As was the case with the boat object under similar conditions (see Graph 8.3), the Best-First traversal combined with the Balanced, Max-Info, and Min-Info tree designs constituted a region of high reliability and efficiency, with the Min-Info combination once again having both the highest $\bar{R}_1$ and $\bar{E}_1$. The Best-First traversal combined with the Random2 tree design was also very efficient, although its reliability was poor.

8.4.2 4 DOF Cluttered Scenes

The cluttered scene results were plotted in Figure 8.17 and the numerical results tabulated in Tables B.16, B.17, and B.18 of Appendix B.6. While the data were in the same arrangement as the isolated results, the efficiencies were somewhat lower. This is explainable, as more extensive traversals would be required on a more varied and deceptive background. The reliability was also slightly lower, which was understood to be an increase in the rate of false positives caused by the more varied background.

In the lower range of the data, plotted in Figure 8.18, the $\bar{E}_1$ values for the Best-First traversal with Balanced, Max-Info, and Min-Info tree designs were roughly the same as in the isolated case. For the Min-Info tree, the $\bar{R}_1$

(a) image

(b) solution

(c) image, rotated view

(d) solution, rotated view

(BF/r3, $E_1 = 676$, $t = 0.18$ sec, $r^+ = 0.71$, n sols = 3)

Figure 8.16: Duck, 4 dof, Cluttered Scene

Figure 8.17: Efficiency ($\bar{E}_1$) vs. Reliability ($\bar{R}_1, \bar{R}_2$), (duck/4 DOF/cluttered)

Figure 8.18: Efficiency ($\bar{E}_1$) vs. Reliability ($\bar{R}_1, \bar{R}_2$), (duck/4 DOF/cluttered)

value decreased. For the Balance and Max-Info trees, the $\bar{R}_1$ value increased over those of the isolated trials. This was attributed to the quality of the particular image sets used.

When compared with the tests of the boat object under cluttered scene conditions (see Figures 8.5 and 8.6), the traversal/tree combinations were once again in a similar arrangement. The Best-First traversal coupled with the random tree designs are off to the right of the plot, with an extremely high efficiency and moderate reliability. The Best-First traversal with the Balanced, Max-Info, and Min-Info trees lie in the mid range with moderately high $\bar{E}_1$ and $\bar{R}_1$. In both the boat and duck trials, the use of the Balanced tree was less efficient than either the Max-Info or Min-Info trees with the Best-First traversal. In the case of the duck, the Balanced tree was not the most reliable of the three.

8.4.3 6 DOF Isolated Scenes

The similarities between the boat and duck results were even more striking in the 6 dof trials. In the higher ranges (Figures 8.20, 8.8) the arrangement of the 4 most reliable combinations was identical, and the efficiency measures were also similar. The reliability of the ultra-efficient Best-First/Min-Info combination was lower for the duck than the boat trials, although the $\bar{R}_1$ of the Best-First/Balance and Best-First/Max-Info combinations were higher

(a) image

(b) solution

(c) image, rotated view

(d) solution, rotated view

(BF/max, $E_1 = 4612$, $t = 3.5$ sec, $r^+ = 0.69$, n sols = 3)

Figure 8.19: Duck, 6 dof, Isolated Scene

Figure 8.20: Efficiency ($\bar{E}_1$) vs. Reliability ($\bar{R}_1, \bar{R}_2$), (duck/6 DOF/isolated)

Figure 8.21: Efficiency ($\bar{E}_1$) vs. Reliability ($\bar{R}_1, \bar{R}_2$), (duck/6 DOF/isolated)

for the duck trials. The Best-First/Balance combination was notable with an efficiency of $\bar{E}_1 = 4100$ and reliability $\bar{R}_1 = 0.75$.

In the lower range of the duck trial results (Figure 8.21) the arrangement was also very similar to that of the boat (Figure 8.9). Here, the N-Error/Balance combination had a remarkably high reliability of $\bar{R}_1 = 0.8$ with a moderately high $\bar{E}_1 = 329$.

8.4.4 6 DOF Cluttered Scenes

An example of a successful trial for the 6 dof duck object under cluttered scene conditions was illustrated in Figure 8.4.4. As in previous cases, the cluttered scene results (Figures 8.23, 8.24) closely matched those of the isolated scenes, with similar efficiencies and slightly lower reliabilities.

(a) image

(b) solution

(c) image, rotated view

(d) solution, rotated view

(BF/max, $E_1 = 4593$, $t = 5.1$ sec, $r^+ = 0.66$, n sols = 12)

Figure 8.22: Duck, 6 dof, Cluttered Scene

Figure 8.23: Efficiency ($\bar{E}_1$) vs. Reliability ($\bar{R}_1, \bar{R}_2$), (duck/6 DOF/cluttered)

Figure 8.24: Efficiency ($\bar{E}_1$) vs. Reliability ($\bar{R}_1, \bar{R}_2$), (duck/6 DOF/cluttered)

8.5 Noise Robustness

An experiment was designed to investigate the degradation in reliability as the noise level increased. We assumed that at the voxel level of detail the effect of noise was to assign erroneous voxel values. The number and distribution of voxels that are affected depends upon the level of noise and the noise model. We assumed that the noise was in a random uniform distribution, and randomly reassigned a specified percentage of the voxels to a particular noise value.

The method was tested for 10 images of the boat under 4 degrees of positional freedom in isolated scene conditions. The Balanced tree design was used in combination with four traversal methods (Best-First, N-Error, Beam, and Heuristic Branch-and-Bound). The reliability metrics $\bar{R}_1$ and $\bar{R}_2$ were plotted as a function of % noise in Figures 8.25 and 8.26. Note that a significant amount of image noise would be required to change the voxel space by 10% at the selected voxel resolution of $\rho = 0.015\text{m}$.

The results indicated that the method was robust to noise level. The reliability degraded gracefully as the noise level increased. As expected, the more extensive traversals were more immune to the effects of noise. The traversals ranked in order of increasing robustness were : Best-First, N-Error, Beam, and Heuristic Branch-and-Bound. This is the same ordinal ranking

Figure 8.25: Noise vs. Reliability ($\bar{R}_1$)

Figure 8.26: Noise vs. Reliability ($\bar{R}_2$)

with respect to increasing computational expense.

8.6 Occlusion Robustness

External occlusion occurs when part of the object of interest is behind some other scene object with respect to the sensor. Another type of external occlusion occurs when the object of interest is partially outside of the sensor's field of view. The effects of occlusion are similar to those of noise in that some fraction of the interrogated voxels will have non-ideal values. Occlusion differs from noise in that, whereas noise will typically affect only isolated voxels or small neighborhoods of adjacent voxels, occlusion can affect a larger connected region.

The sequence of voxels interrogated as a tree is traversed does not necessarily form a connected region, and so the robustness of the method to occlusion is considered to be similar to those of noise. An isolated example where the method was successful in a scene with non-trivial levels of occlusion is illustrated in Figure 8.27. Although anecdotal, these tests support the claim that the method is robust to occlusions.

(a) image

(b) solution

(c) image, rotated view

(d) solution, rotated view

(BF/bal, $E_1 = 95$, $t = 0.41$ sec, $r^+ = 0.81$, n sols = 4)

Figure 8.27: Boat, 4 dof, Occluded Scene

8.7 Cascading for Enhanced Reliability

A technique called *cascading* can be used to enhance reliability. Cascading is the process of traversing a series of trees for each seed, and returning the single best result from the group. It is based on the observation that different traversal/tree combinations fail differently.

Assume that two particular combinations were tested over a positive image set and were both found to have reliabilities of $\bar{R}_2 = 0.50$. Assume further that the subsets of successful trials for each combination were mutually exclusive, i.e. the successful trials for one combination were the failed trials for the other. If the results were cascaded, then the pooled reliability would be $\bar{R}_1 = 1.0$, albeit with a reduced pooled efficiency.

As an example taken from real data, consider the experiment of the boat model with a 6 dof pose space under cluttered scene conditions (as described in Section 8.3.5). The measured reliability and efficiency values of Best-First traversal with the Balance, Max-Info and Min-Info tree designs for each of the 20 image trials were tabulated in Table 8.7. The second column of the table reports a string of 20 binary values which indicated the R_1 value for each trial. It can be seen that, among the 6 failures of the Balanced tree trials and the 7 failures of the Max-Info trials, only 3 were in common. Also, the single success of the Min-Info trial is unique.

<i>trav.</i>	R_1	$\bar{R}_1$	$\bar{R}_2$	$\bar{E}_1$
bal	11010111100111111010	0.70	0.653	3686
max	11111110001100111010	0.65	0.642	4049
min	00000000000000000001	0.05	0.04	28800

Table 8.7: Performance Without Cascading

The results from two different cascades of the above tests, one with the Balance and Max-Info trees and one with all 3 trees, were tabulated in Table 8.8. Note that the R_1 string was not the logical OR of the component strings. This is because the results from each seed was taken to be the traversal result with the highest R_2 value, which may or may not be the a correct match. (In the prior example, for the cascaded reliability to be unity, an additional requirement would be that $R_{2i} > R_{2j} \leftrightarrow R_{1i} > R_{1j}$.) In the final three columns of Table 8.8 it can be seen that there was an increase in the reliability metrics $\bar{R}_1$ and $\bar{R}_2$ and a corresponding decrease in the pooled efficiency $\bar{E}_{1p}$ which is defined as :

$$\bar{E}_{1p} = \frac{1}{\sum 1/\bar{E}_{1i}} \quad (8.4)$$

It is interesting to note that while the Min-Info tree on its own was very unreliable, it succeeded in a trial (image 20) were the other 2 trees failed,

<i>trav.</i>	R_1	$\bar{R}_1$	$\bar{R}_2$	$\bar{E}_{1p}$
bal+max	1111111100110111010	0.75	0.67	1929
bal+max+min	1111111100110111011	0.80	0.67	1808

Table 8.8: Performance With Cascading

and therefore improved reliability when included in the cascade. As it was an extremely efficient traversal, the resultant reduction in pooled efficiency was relatively small.

8.8 Parallelism for Enhanced Efficiency

The method is easily parallelized by assigning a separate processor resource for each seed voxel. As there are typically thousands of seed voxels per image, the method is potentially massively parallelizable.

The parallel method was organized as a resource pool of multiple *compute* processes, coordinated by a single *control* process. Pseudocode for both the control and compute processes are illustrated in Figure 8.28. At startup each compute process, of which one existed per processor, was initialized with a local copy of the voxel space and the tree. During recognition, the single control process allocated seed voxels in turn to each available compute process. In this way, for any single compute process, the tree was traversed

for only a fraction of the total number of seeds.

The method was implemented and tested on a four processor platform using PVM for interprocess communication. The time performance as a function of the number of processors was illustrated in Graph 8.29. The less than linear speedup is attributed in part to the overheads of PVM on the particular computing platform used. Another potential contributing factor was the reliance of the Best-First traversal on the estimate of maximum correlation to prune subtrees (see Chapter 7). Even when communicated across compute nodes (as indicated in the pseudocode of Figure 8.28), the progression of this value would differ from the single processor case.

It is believed that, with a more efficient interprocess communication mechanism, the performance enhancement with respect to the number of processors could be made nearly linear for up to tens of processors.

8.9 Summary of Chapter

In this Chapter, a series of experiments were described which demonstrated the reliability and efficiency of the method under various conditions. The methodology was to test all combinations of tree traversal and tree design method over sets of 20 images in isolated and cluttered conditions. The experiments were also categorized according to the dimension of the object's

```

compute_process()
{
    load tree ;
    load voxelspace ;

    while (!done)
    {
        send id to control process ;
        blocking receive seed from control process ;
        traverse(seed,tree,voxelspace) ;
    }
}

control_process()
{
    load voxelspace ;

    seed = get_next(voxelspace)

    while (seed != NULL)
    {
        blocking receive id from available compute process ;
        send seed to available compute process ;
        seed = get_next(voxelspace)
    }
}

```

Figure 8.28: Parallel Implementation Pseudocode

Figure 8.29: Number of Processors (m) vs. Time (t [sec.])

pose space. In total, approximately 160 images were tested.

Overall, the method was shown to perform well. The combination of the Best-First traversal with the Min-Info, Max-Info, and Balanced tree design consistently exhibited moderately high efficiency and reliability ratings. In the case of isolated scenes, the Best-First traversal with the Min-Information design was extremely efficient and moderately reliable, although in the cluttered scenes, the reliability of this combination reduced.

An interesting and significant observation from these tests was that the results were not spurious. Examining the efficiency/reliability plots, it was

observed that the various traversal/design combinations performed approximately the same relative to each other across all test cases. This was true across not only different pose space dimensions for the same objects, but across different objects as well. This indicates that the results are not specific to the objects or images used in these test, but are general.

A statistical t-test was applied to trials of the boat object under 4 dof conditions. A value of the correlation threshold was determined which significantly discriminated between true and false positive matches. An ANOVA test was also applied to same case, and it was concluded that certain tree traversal/design combinations had significantly different performance characteristics. Fisher's LSD was subsequently used to cluster the results.

Further experiments demonstrated the robustness of the method to noise and occlusion. A technique for enhancing reliability by cascading traversals was described. Finally, the time performance improvement of a parallel version of method was demonstrated and tested.

Chapter 9

Conclusions

There are a number of open questions and research issues which follow from this work;

- **multiple object database:** While the presented experimentation was limited to the case of single objects, the method extends directly to the recognition of multiple scene objects. There is no reason why the templates encoded in a tree need represent a single object. The only requirement would seem to be that the objects be of roughly the same radius, so that a common voxel resolution can be selected. One future research issue is to determine the point at which it is no longer effective to continue to encode additional templates into a single tree.
- **improved error model:** We used a simple black-box error model based upon the *fallible oracle* assumption. A better understanding of

the sources of error could lead to more efficient and reliable traversal.

- **verification:** In setting a value for the correlation threshold, our strategy was to err in favor of false positives. It would be useful to have a way to return the accurate solutions. One possibility is to base the verification on the constraint functions (equations 3.1 and 3.2) which were proposed in the definition of the recognition problem.
- **enhanced decision rule:** The decision rule which was evaluated at each internal node during tree traversal was based upon a binary point probe, and it considered only a single point. There is no reason why additional information, such as intensity, could not be included in the decision rule function.
- **template set reduction:** The template set for a given model was based upon a regular tessellation of the viewing sphere. It may be possible to represent an entire ICP potential well with only one template. As the well may in practice be quite large, this could greatly reduce the size of the template set and improve both efficiency and reliability.
- **multi-resolution:** Geometry probing is inherently a multi-resolution strategy, as only the currently relevant data is admitted. It may be useful to have a hierarchy of voxel space resolutions, rather than se-

lecting a single resolution. This could allow the encoding of multiple variable scale objects into single tree.

- **application to intensity images:** With minor modifications, the method could be applied to the efficient matching of a set of 2D intensity image templates.
- **general expected traversal cost:** In the decision tree literature, it seems that expressions for the expected traversal cost have been developed solely for the case of depth-first non-tentative traversal. One wonders whether these results generalize to other traversal methods. It would be particularly useful to determine whether there exist tractable methods to construct optimal decision trees for other traversal methods.
- **binary point probe strategy:** The binary point probe model proposed in this work is particularly useful for the practical case of dense range data. It has not been encountered in any other work on Geometric Probing. It would be interesting to relate this probe model with the theoretical results developed for other probe models.

The research has shown that object recognition is a complex task. It is unlikely that any single method will be sufficient to solve all cases. The

most general and robust solutions will therefore likely be a combination of a number of techniques. To this end, it would be necessary to have a framework to empirically compare the behaviors of alternate methods. The experimental methodology presented in this work is one possible beginning for such a comparison.

Appendix A

Detailed Statistical Calculations

A.1 Correlation Threshold Determination, $\rho = 0.015$

This section details the calculations described in Section 8.3.1 for the case $\rho = 0.015$, and follows the t-test procedure described by Ott [104, p 143].

The objective was to determine whether the maximum correlation values r^+ were significantly different between positive and negative trials. The data from these trials were summarized in Table 8.1. Here, subscript 1 represents the positive trial, and subscript 2 represents the negative trial.

The null hypothesis (H_0), alternative hypothesis (H_a), test statistic (T.S.), and rejection region (R.R.) are as follows ;

$$H_0 : r_1^+ - r_2^+ = 0 \quad (\text{A.1})$$

$$H_a : r_1^+ - r_2^+ > 0 \quad (\text{A.2})$$

$$T.S. : t' = \frac{\bar{r}_1^+ - \bar{r}_2^+}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}} \quad (\text{A.3})$$

$$R.R. : \text{reject } H_0 \text{ if } t' > t_\alpha \quad (\text{A.4})$$

The degrees of freedom of the t-test are calculated as ;

$$c = \frac{s_1^2/n_1}{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}} \quad (\text{A.5})$$

$$df = \frac{(n_1 - 1)(n_2 - 1)}{(n_2 - 1)c^2 + (1 - c)^2(n_1 - 1)} \quad (\text{A.6})$$

Substituting the values from Table 8.1 gives ;

$$c = 0.733 \quad (\text{A.7})$$

$$df = 31.6 \quad (\text{A.8})$$

$$t' = 0.965 \quad (\text{A.9})$$

At $df = 30$, the t-distribution for $\alpha = 0.10$ has a value of $t_{0.10} = 1.31$. We therefore cannot reject H_0 , and conclude that there is no evidence at this level of significance that the positive and negative trials were different.

A.2 Correlation Threshold Determination, $\rho = 0.010$

This section details the calculations described in Section 8.3.1 for the case $\rho = 0.015$, and repeats the t-test procedure described in Section A.1. The data is summarized in Table 8.2. H_0 , H_a , T.S., and R.R. are as in equations A.1 to A.4.

$$c = 0.460 \quad (\text{A.10})$$

$$df = 40.0 \quad (\text{A.11})$$

$$t' = 1.647 \quad (\text{A.12})$$

At $df = 40$, the t-distribution for $\alpha = 0.10$ has a value of $t_{0.10} = 1.303$. We therefore reject H_0 in favor of H_a as $t' > t_{0.10}$. We conclude that the positive trial correlation results were significantly larger than the negative trial results.

A.3 R_2 ANOVA Calculations

A.3.1 ANOVA

This test follows the procedure described in Ott [104, p 331]. The objective was to determine whether the maximum correlation values differed significantly for various samples of tree/traversal combinations, as described in

Section 8.3.1 The results from the experiments were tabulated in Table A.1, which corresponds to Table 10.4 of Ott [104, p 331].

The quantities generated from this table are as follows :

$$TSS = \sum_{i,j} y_{i,j}^2 - \frac{G^2}{n} = 26.4 \quad (\text{A.13})$$

$$SSB = \frac{\sum T_i^2}{n_i} - \frac{G^2}{n} = 123.4 \quad (\text{A.14})$$

$$SSW = TSS - SSB = 18.5 \quad (\text{A.15})$$

$$s_B^2 = \frac{SSB}{t-1} = 0.342 \quad (\text{A.16})$$

$$s_W^2 = \frac{SSW}{n-t} = 0.0858 \quad (\text{A.17})$$

$$F = \frac{s_B^2}{s_W^2} = 3.98 \quad (\text{A.18})$$

For $df_1 = 23$ and $df_2 = 216$, at a significance level of $\alpha = 0.1$, the null hypothesis H_0 that all of the R_2 are equal is rejected if $F > 2.17$, which indeed it is. We therefore reject H_0 and continue on to Fisher's LSD test.

A.3.2 Fisher's LSD

This test follows the procedure described in Ott [104, p 366]. The objective is to find the *Least Significant Difference (LSD)*, which is the smallest value of R_2 by which two samples can be said to differ at a certain level of significance.

#	Tree	Trav.	Image										$\bar{r}^+$
			1	2	3	4	5	6	7	8	9	10	
1	bal	beam	1.0	.96	.75	.83	.85	.77	.91	.84	.84	.90	.867
2	max	beam	1.0	.97	.75	.83	.85	.80	.91	.80	.84	.90	.865
3	min	beam	.87	.93	.0	.83	0	.80	0	0	.84	0	.427
4	rand1	beam	1.0	.78	0	0	.82	0	.80	0	0	.87	.428
5	rand2	beam	.97	.93	0	0	.83	.76	.78	0	0	.83	.509
6	rand3	beam	.84	.97	.75	0	.80	0	0	.77	0	.83	.495
7	bal	best 1 st	.94	.97	0	.83	.89	.79	.91	.80	.84	.82	.78
8	max	best 1 st	1.0	.90	.75	.78	.78	.77	0	0	.84	.89	.672
9	min	best 1 st	1.0	.97	0	.83	.89	.77	.91	.84	.79	.87	.786
10	rand1	best 1 st	1.0	.85	0	.83	.82	0	.80	0	0	.77	.508
11	rand2	best 1 st	.84	.93	0	0	.80	0	.81	0	0	.83	.421
12	rand3	best 1 st	.84	.97	0	0	.80	0	0	0	0	.87	.347
13	bal	BnB	1.0	.97	.75	.83	.89	.80	.91	.84	.84	.90	.873
14	max	BnB	1.0	.97	.75	.83	.82	.80	.91	.84	.84	.90	.867
15	min	BnB	1.0	.97	0	.83	.89	.78	.91	.84	.84	.90	.795
16	rand1	BnB	1.0	.97	.75	.83	.89	.80	.91	.84	.84	.90	.873
17	rand2	BnB	1.0	.97	.75	.83	.89	.80	.91	.84	.84	.90	.873
18	rand3	BnB	1.0	.97	.75	.83	.89	.80	.91	.84	.84	.90	.873
19	bal	N err	.95	.97	.75	.83	.89	.79	.91	.80	.84	.82	.854
20	max	N err	1.0	.97	.75	.78	.78	.77	.91	.80	.84	.90	.851
21	min	N err	1.0	.97	0	.83	.89	.78	.91	.84	.84	.90	.795
22	rand1	N err	1.0	.97	0	.83	.89	.77	.91	.77	.84	.90	.787
23	rand2	N err	1.0	.97	.75	.83	.82	.80	.91	.84	.82	.89	.863
24	rand3	N err	1.0	.97	0	.83	.89	.80	.91	.84	.84	.90	.797

Table A.1: R_2 ANOVA Data

The expression for the LSD with common sample size n is :

$$LSD = t_{\alpha/2} \sqrt{\frac{s_W^2}{n}} \quad (\text{A.19})$$

A level of significance of $\alpha = 0.05$ with $df = 216$ gives $t_{\alpha/2} = 1.96$, which yields a value of $LSD = 0.257$.

A.4 E_1 ANOVA Calculations

A.4.1 ANOVA

The objective of this test was to determine whether the values of E_1 differed significantly for various samples of tree/traversal combinations, as described in Section 8.3.2.2. The results from the experiments were tabulated in Table A.2, which corresponds to Table 10.4 of Ott [104, p 331].

The quantities generated from this table are ;

$$TSS = \sum_{i,j} y_{i,j}^2 - \frac{G^2}{n} = 5,574,306 \quad (\text{A.20})$$

$$SSB = \frac{\sum T_i^2}{n_i} - \frac{G^2}{n} = 4,530,669 \quad (\text{A.21})$$

$$SSW = TSS - SSB = 1,043,637 \quad (\text{A.22})$$

$$s_B^2 = \frac{SSB}{t-1} = 205,939 \quad (\text{A.23})$$

#	Tree	Trav.	Image										$\bar{E}_1$
			1	2	3	4	5	6	7	8	9	10	
1	bal	beam	7.9	8.0	7.8	7.9	7.8	7.9	7.8	7.9	8.0	7.8	7.88
2	max	beam	5.5	5.5	5.5	5.6	5.5	5.6	5.5	5.5	5.6	5.4	5.53
3	min	beam	35.4	34.4	44.2	41.9	43.8	43.5	42.8	47.2	57.3	60.0	45.0
4	rand1	beam	26.7	29.0	37.3	38.7	35.0	36.8	38.4	35.7	37.7	38.2	35.4
5	rand2	beam	30.4	34.0	35.2	34.2	35.7	32.6	34.1	37.0	31.3	34.2	33.9
6	rand3	beam	22.6	22.3	22.9	23.0	22.7	22.8	22.9	23.3	23.4	22.4	22.8
7	bal	best 1 st	103	99.0	92.6	93.7	93.4	94.4	93.0	9.5	95.1	93.0	95.0
8	max	best 1 st	177	175	163	172	160	167	173	181	190	156	171
9	min	best 1 st	179	178	177	177	179	178	179	178	174	174	178
10	rand1	best 1 st	462	330	286	236	314	236	241	258	1110	445	392
11	rand2	best 1 st	268	307	563	733	460	815	718	688	888	445	588
12	rand3	best 1 st	1138	1323	2634	5070	6700	6700	14175	6705	13320	2820	6059
13	bal	BnB	2.8	3.1	3.3	4.1	4.8	3.5	4.5	4.0	5.3	6.2	4.2
14	max	BnB	2.7	3.4	3.0	3.6	4.4	3.3	4.2	3.6	5.2	6.5	4.0
15	min	BnB	3.5	3.8	6.3	6.3	6.4	6.2	6.6	6.5	6.5	6.7	5.9
16	rand1	BnB	4.7	5.6	8.2	9.3	8.6	8.9	9.0	10.2	10.1	9.6	8.4
17	rand2	BnB	5.0	5.9	9.2	9.8	9.3	9.1	9.4	9.9	10.1	10.3	8.8
18	rand3	BnB	5.1	6.0	6.3	8.8	8.0	7.9	8.4	9.3	9.2	8.4	7.7
19	bal	N err	16.4	17.0	18.3	18.1	18.2	18.1	18.7	18.5	17.4	17.7	17.8
20	max	N err	18.2	18.0	16.8	17.2	17.3	16.7	17.0	17.8	14.5	15.4	16.9
21	min	N err	3.4	3.7	2.3	2.3	2.0	2.3	1.9	2.1	1.8	1.6	2.4
22	rand1	N err	13.5	16.3	9.4	12.6	11.7	12.5	10.7	9.6	6.9	8.6	11.2
23	rand2	N err	11.8	12.9	8.4	9.4	9.2	9.4	8.6	8.0	6.3	7.7	9.2
24	rand3	N err	12.0	10.4	8.8	8.1	8.9	8.4	8.5	8.1	7.9	8.7	9.0

Table A.2: E_1 ANOVA Data

$$s_W^2 = \frac{SSW}{n - t} = 4,809 \quad (\text{A.24})$$

$$F = \frac{s_B^2}{s_W^2} = 42.8 \quad (\text{A.25})$$

Note that the combination of the Best-First Traversal with the Random-3 tree (trial 12) was not excluded from the above computations.

For $df_1 = 23$ and $df_2 = 216$, at a significance level of $\alpha = 0.10$, the null hypothesis H_0 that all of the E_1 values are equal is rejected if $F > 2.17$, which indeed it is. We therefore reject H_0 and continue on to Fisher's LSD test.

A.4.2 Fisher's LSD

This test follows the procedure described in Ott [104, p 366] and applied in Section A.3.2. The objective is to find the LSD, which is the smallest value of E_1 by which two samples can be said to differ at a certain level of significance.

The expression for the LSD with common sample size n is :

$$LSD = t_{\alpha/2} \sqrt{\frac{s_W^2}{n}} \quad (\text{A.26})$$

A level of significance of $\alpha = 0.05$ with $df = 216$ gives $t_{\alpha/2} = 1.96$, which yields a value of $LSD = 60.8$.

A.5 Rejection Region Boundaries, $\rho = 0.015$

This section details the calculations of the boundaries of the rejection regions described in Section 8.3.1 for the case $\rho = 0.015$. The calculation follows the t-test procedure described by Ott [104, p 126].

A.5.1 Positive Trial

The objective was to determine the lower boundary R_2^- of the rejection region of the t-test Null Hypothesis. This lower boundary represents the smallest value of R_2 which can be expected for a true positive match, with the probability of a false negative error fixed at $\alpha = 0.10$.

The t-test is organized as follows :

$$H_0 : R_2 = R_2^- \quad (\text{A.27})$$

$$H_a : R_2 > R_2^- \quad (\text{A.28})$$

$$T.S. : t = \frac{\bar{R}_2 - R_2^-}{s\sqrt{n}} \quad (\text{A.29})$$

$$R.R. : t > t_\alpha \quad (\text{A.30})$$

$$(\text{A.31})$$

Equation A.30 can be rewritten as :

$$R_2^- < \bar{R}_2 - t_\alpha \frac{s}{\sqrt{n}} \quad (\text{A.32})$$

For dof= 19 and $\alpha = 0.10$, $t_\alpha = 1.328$. Substituting the values of Table 8.1 yields a value of $R_2^- < 0.8231$.

A.5.2 Negative Trial

The objective was to determine the upper boundary R_2^+ of the rejection region of the t-test Null Hypothesis. This upper boundary represents the largest value of R_2 which can be expected for a true negative match, with a probability of a false negative error fixed at $\alpha = 0.10$.

The t-test is organized as follows :

$$H_0 : R_2 = R_2^- \quad (\text{A.33})$$

$$H_a : R_2 < R_2^- \quad (\text{A.34})$$

$$T.S. : t = \frac{\bar{R}_2 - R_2^-}{s\sqrt{n}} \quad (\text{A.35})$$

$$R.R. : t < -t_\alpha \quad (\text{A.36})$$

$$(\text{A.37})$$

Equation A.36 can be rewritten as :

$$R_2^- > \bar{R}_2 + t_\alpha \frac{s}{\sqrt{n}} \quad (\text{A.38})$$

For dof= 21 and $\alpha = 0.10$, $t_\alpha = 1.328$. Substituting the values of Table 8.1 yields a value of $R_2^+ > 0.8373$.

A.6 Rejection Region Boundaries, $\rho = 0.010$

This section follows the procedure described in Section A.6 for the case $\rho = 0.010$.

A.6.1 Positive Trial

Equation A.30 can be rewritten as :

$$R_2^- < \bar{R}_2 - t_\alpha \frac{s}{\sqrt{n}} \quad (\text{A.39})$$

For dof= 19 and $\alpha = 0.10$, $t_\alpha = 1.328$. Substituting the values of Table 8.2 yields a value of $R_2^- < 0.7735$.

A.6.2 Negative Trial

Equation A.36 can be rewritten as :

$$R_2^- > \bar{R}_2 + t_\alpha \frac{s}{\sqrt{n}} \quad (\text{A.40})$$

For dof= 19 and $\alpha = 0.10$, $t_\alpha = 1.328$. Substituting the values of Table 8.2 yields a value of $R_2^+ > 0.7697$.

Appendix B

Tabulated Test Results

B.1 Boat/4 DOF/Isolated

	<i>tree</i>					
<i>traversal</i>	<i>Bal</i>	<i>Max-Info</i>	<i>Min-Info</i>	<i>Rand-1</i>	<i>Rand-2</i>	<i>Rand-3</i>
Best-First	95	171	178	392	588	6059
Beam	8	6	45	35	34	23
N-Error	18	17	2	11	9	9
Branch-n-Bound	4	4	6	8	9	8

Table B.1: $\bar{E}_1$ Results (boat/4dof/isolated)

	<i>tree</i>					
<i>traversal</i>	<i>Bal</i>	<i>Max-Info</i>	<i>Min-Info</i>	<i>Rand-1</i>	<i>Rand-2</i>	<i>Rand-3</i>
Best-First	.90	.80	.90	.60	.50	.40
Beam	1	1	.50	.50	.60	.60
N-Error	1	1	.90	.90	1	.90
Branch-n-Bound	1	1	.90	1	1	1

Table B.2: $\bar{R}_1$ Results (boat/4dof/isolated)

	<i>tree</i>					
<i>traversal</i>	<i>Bal</i>	<i>Max-Info</i>	<i>Min-Info</i>	<i>Rand-1</i>	<i>Rand-2</i>	<i>Rand-3</i>
Best-First	.77	.67	.79	.51	.42	.35
Beam	.87	.87	.43	.43	.51	.49
N-Error	.85	.85	.80	.79	.86	.80
Branch-n-Bound	.87	.87	.80	.87	.87	.87

Table B.3: $\bar{R}_2$ Results (boat/4dof/isolated)

B.2 Boat/4 DOF/Cluttered

	<i>tree</i>					
<i>traversal</i>	<i>Bal</i>	<i>Max-Info</i>	<i>Min-Info</i>	<i>Rand-1</i>	<i>Rand-2</i>	<i>Rand-3</i>
Best-First	95	162	177	309	389	2329
Beam	8	6	46	36	36	23
N-Error	17	17	3	12	10	10
Branch-n-Bound	6	6	6	9	9	8

Table B.4: $\bar{E}_1$ Results (boat/4dof/cluttered)

	<i>tree</i>					
<i>traversal</i>	<i>Bal</i>	<i>Max-Info</i>	<i>Min-Info</i>	<i>Rand-1</i>	<i>Rand-2</i>	<i>Rand-3</i>
Best-First	.95	.80	.75	.70	.55	.35
Beam	.95	.95	.65	.70	.60	.50
N-Error	.95	.85	.90	.80	.90	.95
Branch-n-Bound	.95	.95	.90	.90	.95	.95

Table B.5: $\bar{R}_1$ Results (boat/4dof/cluttered)

	<i>tree</i>					
<i>traversal</i>	<i>Bal</i>	<i>Max-Info</i>	<i>Min-Info</i>	<i>Rand-1</i>	<i>Rand-2</i>	<i>Rand-3</i>
Best-First	.88	.75	.75	.68	.51	.44
Beam	.89	.89	.59	.63	.57	.51
N-Error	.89	.80	.85	.76	.85	.88
Branch-n-Bound	.89	.89	.85	.88	.89	.89

Table B.6: $\bar{R}_2$ Results (boat/4dof/cluttered)

B.3 Boat/6 DOF/Isolated

	<i>tree</i>		
<i>traversal</i>	<i>Bal</i>	<i>Max-Info</i>	<i>Min-Info</i>
Best-First	3628	4175	25828
Beam	257	164	9504
N-Error	309	300	13
Branch-n-Bound	6	5	56

Table B.7: $\bar{E}_1$ Results (boat/6dof/isolated)

	<i>tree</i>		
<i>traversal</i>	<i>Bal</i>	<i>Max-Info</i>	<i>Min-Info</i>
Best-First	.60	.35	.60
Beam	.45	.70	0
N-Error	.60	.40	1
Branch-n-Bound	.95	.95	1

Table B.8: $\bar{R}_1$ Results (boat/6dof/isolated)

	<i>tree</i>		
<i>traversal</i>	<i>Bal</i>	<i>Max-Info</i>	<i>Min-Info</i>
Best-First	.63	.65	.42
Beam	.67	.68	0
N-Error	.66	.66	.80
Branch-n-Bound	.80	.80	.80

Table B.9: $\bar{R}_2$ Results (boat/6dof/isolated)

B.4 Boat/6 DOF/Cluttered

	<i>tree</i>		
<i>traversal</i>	<i>Bal</i>	<i>Max-Info</i>	<i>Min-Info</i>
Best-First	3686	4049	28800
Beam	261	178	9456
N-Error	315	313	18
Branch-n-Bound	8	6	29

Table B.10: $\bar{E}_1$ Results (boat/6dof/cluttered)

	<i>tree</i>		
<i>traversal</i>	<i>Bal</i>	<i>Max-Info</i>	<i>Min-Info</i>
Best-First	.70	.65	.05
Beam	.55	.65	0
N-Error	.65	.75	1
Branch-n-Bound	1	1	1

Table B.11: $\bar{R}_1$ Results (boat/6dof/cluttered)

	<i>tree</i>		
<i>traversal</i>	<i>Bal</i>	<i>Max-Info</i>	<i>Min-Info</i>
Best-First	.65	.64	.04
Beam	.68	.66	0
N-Error	.67	.66	.79
Branch-n-Bound	.80	.80	.79

Table B.12: $\bar{R}_2$ Results (boat/6dof/cluttered)

B.5 Duck/4 DOF/Isolated

	<i>tree</i>					
<i>traversal</i>	<i>Bal</i>	<i>Max-Info</i>	<i>Min-Info</i>	<i>Rand-1</i>	<i>Rand-2</i>	<i>Rand-3</i>
Best-First	116	185	224	-	2679	916
Beam	9	8	72	57	100	51
N-Error	21	23	2	9	9	6
Branch-n-Bound	3	2	10	7	5	6

Table B.13: $\bar{E}_1$ Results (duck/4dof/isolated)

	<i>tree</i>					
<i>traversal</i>	<i>Bal</i>	<i>Max-Info</i>	<i>Min-Info</i>	<i>Rand-1</i>	<i>Rand-2</i>	<i>Rand-3</i>
Best-First	.60	.65	.75	0	.35	.60
Beam	.95	.90	.15	.10	.10	.20
N-Error	.65	.90	.90	1	1	.85
Branch-n-Bound	1	1	.95	1	1	1

Table B.14: $\bar{R}_1$ Results (duck/4dof/isolated)

	<i>tree</i>					
<i>traversal</i>	<i>Bal</i>	<i>Max-Info</i>	<i>Min-Info</i>	<i>Rand-1</i>	<i>Rand-2</i>	<i>Rand-3</i>
Best-First	.57	.62	.61	0	.33	.47
Beam	.77	.76	.16	.21	.23	.30
N-Error	.58	.71	.74	.77	.78	.68
Branch-n-Bound	.79	.80	.74	.80	.80	.79

Table B.15: $\bar{R}_2$ Results (duck/4dof/isolated)

B.6 Duck/4 DOF/Cluttered

	<i>tree</i>					
<i>traversal</i>	<i>Bal</i>	<i>Max-Info</i>	<i>Min-Info</i>	<i>Rand-1</i>	<i>Rand-2</i>	<i>Rand-3</i>
Best-First	117	174	224	-	1233	627
Beam	10	8	67	55	93	49
N-Error	20	20	3	11	10	8
Branch-n-Bound	3	3	9	7	7	7

Table B.16: $\bar{E}_1$ Results (duck/4dof/cluttered)

	<i>tree</i>					
<i>traversal</i>	<i>Bal</i>	<i>Max-Info</i>	<i>Min-Info</i>	<i>Rand-1</i>	<i>Rand-2</i>	<i>Rand-3</i>
Best-First	.76	.81	.48	0	.24	.57
Beam	.95	.90	.05	.43	.14	.19
N-Error	.86	.86	.76	.90	.90	.86
Branch-n-Bound	.95	.95	.86	.95	.95	.95

Table B.17: $\bar{R}_1$ Results (duck/4dof/cluttered)

	<i>tree</i>					
<i>traversal</i>	<i>Bal</i>	<i>Max-Info</i>	<i>Min-Info</i>	<i>Rand-1</i>	<i>Rand-2</i>	<i>Rand-3</i>
Best-First	.70	.75	.64	0	.42	.53
Beam	.76	.76	.13	.49	.37	.40
N-Error	.73	.75	.77	.79	.78	.77
Branch-n-Bound	.82	.82	.78	.82	.82	.82

Table B.18: $\bar{R}_2$ Results (duck/4dof/cluttered)

B.7 Duck/6 DOF/Isolated

	<i>tree</i>		
<i>traversal</i>	<i>Bal</i>	<i>Max-Info</i>	<i>Min-Info</i>
Best-First	4100	4531	28800
Beam	295	141	9387
N-Error	329	271	33
Branch-n-Bound	4	4	29

Table B.19: $\bar{E}_1$ Results (duck/6dof/isolated)

	<i>tree</i>		
<i>traversal</i>	<i>Bal</i>	<i>Max-Info</i>	<i>Min-Info</i>
Best-First	.75	.55	.10
Beam	.60	.50	0
N-Error	.80	.55	.75
Branch-n-Bound	.75	.75	.80

Table B.20: $\bar{R}_1$ Results (duck/6dof/isolated)

	<i>tree</i>		
<i>traversal</i>	<i>Bal</i>	<i>Max-Info</i>	<i>Min-Info</i>
Best-First	.62	.61	.10
Beam	.61	.63	.06
N-Error	.62	.65	.71
Branch-n-Bound	.72	.71	.72

Table B.21: $\bar{R}_2$ Results (duck/6dof/isolated)

B.8 Duck/6 DOF/Cluttered

	<i>tree</i>		
<i>traversal</i>	<i>Bal</i>	<i>Max-Info</i>	<i>Min-Info</i>
Best-First	4058	4586	28800
Beam	295	159	9173
N-Error	321	302	31
Branch-n-Bound	6	5	24

Table B.22: $\bar{E}_1$ Results (duck/6dof/cluttered)

	<i>tree</i>		
<i>traversal</i>	<i>Bal</i>	<i>Max-Info</i>	<i>Min-Info</i>
Best-First	.55	.45	.10
Beam	.65	.55	0
N-Error	.55	.45	0.70
Branch-n-Bound	.70	.70	.65

Table B.23: $\bar{R}_1$ Results (duck/6dof/cluttered)

	<i>tree</i>		
<i>traversal</i>	<i>Bal</i>	<i>Max-Info</i>	<i>Min-Info</i>
Best-First	.56	.59	.15
Beam	.66	.62	.06
N-Error	.60	.61	.70
Branch-n-Bound	.72	.72	.70

Table B.24: $\bar{R}_2$ Results (duck/6dof/cluttered)

Appendix C

Selected Positive Results

C.1 Boat, 4 DOF, Isolated

(a) image

(b) solution

(BF/r2, $E_1 = 460$, $t = 0.07$ sec, $r^+ = 0.78$, n sols = 2)

Figure C.1: Boat, 4 dof, Isolated Scene

(a) image

(b) solution

(BF/min, $E_1 = 179$, $t = 0.53$ sec, $r^+ = 1.0$, n sols = 83)

Figure C.2: Boat, 4 dof, Isolated Scene

C.2 Boat, 4 DOF, Cluttered

(a) image

(b) solution

(BF/r2, $E_1 = 482$, $t = 0.13$ sec, $r^+ = 0.85$, n sols = 4)

Figure C.3: Boat, 4 dof, Cluttered Scene

(a) image

(b) solution

(BF/r2, $E_1 = 353$, $t = 0.17$ sec, $r^+ = 0.91$, n sols = 5)

Figure C.4: Boat, 4 dof, Cluttered Scene

C.3 Boat, 4 DOF, Occluded

(a) image

(b) solution

(c) image, rotated view

(d) solution, rotated view

(BF/r3, $E_1 = 1560$, $t = 0.07$ sec, $r^+ = 0.75$, n sols = 1)

Figure C.5: Boat, 4 dof, Occluded Scene

C.4 Boat, 6 DOF, Isolated

(a) image

(b) solution

(BF/min, $E_1 = 28396$, $t = 3.4$ sec, $r^+ = 0.70$, n sols = 2)

Figure C.6: Boat, 6 dof, Isolated Scene

(a) image

(b) solution

(BF/min, $E_1 = 3535$, $t = 4.1$ sec, $r^+ = 0.66$, n sols = 14)

Figure C.7: Boat, 6 dof, Isolated Scene

C.5 Boat, 6 DOF, Cluttered

(a) image

(b) solution

(BF/bal, $E_1 = 3678$, $t = 5.1$ sec, $r^+ = 0.72$, n sols = 74)

Figure C.8: Boat, 6 dof, Cluttered Scene

(a) image

(b) solution

(BF/max, $E_1 = 4160$, $t = 3.6$ sec, $r^+ = 0.78$, n sols = 54)

Figure C.9: Boat, 6 dof, Cluttered Scene

C.6 Boat, 6 DOF, Occluded

(a) image

(b) solution

(c) image, rotated view

(d) solution, rotated view

(BF/max, $E_1 = 4395$, $t = 7.2$ sec, $r^+ = 0.74$, n sols = 330)

Figure C.10: Boat, 6 dof, Occluded Scene

C.7 Duck, 4 DOF, Isolated

(a) image

(b) solution

(BF/r3, $E_1 = 1016$, $t = 0.36$ sec, $r^+ = 0.87$, n sols = 13)

Figure C.11: Duck, 4 dof, Isolated Scene

(a) image

(b) solution

(BF/r3, $E_1 = 995$, $t = 0.12$ sec, $r^+ = 0.66$, n sols = 2)

Figure C.12: Duck, 4 dof, Isolated Scene

C.8 Duck, 4 DOF, Cluttered

(a) image

(b) solution

(BF/r3, $E_1 = 526$, $t = 0.28$ sec, $r^+ = 0.93$, n sols = 2)

Figure C.13: Duck, 4 dof, Cluttered Scene

(a) image

(b) solution

(BF/r3, $E_1 = 570$, $t = 0.21$ sec, $r^+ = 0.78$, n sols = 3)

Figure C.14: Duck, 4 dof, Cluttered Scene

C.9 Duck, 4 DOF, Occluded

(BF/bal, $E_1 = 115$, $t = 1.0$ sec, $r^+ = 0.81$, n sols = 31)

Figure C.15: Duck, 4 dof, Occluded Scene

(a) image

(b) solution

(BF/r2, $E_1 = 898$, $t = 0.2$ sec, $r^+ = 0.63$, n sols = 2)

Figure C.16: Duck, 4 dof, Occluded Scene

C.10 Duck, 6 DOF, Isolated

(a) image

(b) solution

(BF/max, $E_1 = 4568$, $t = 3.5$ sec, $r^+ = 0.74$, n sols = 27)

Figure C.17: Duck, 6 dof, Isolated Scene

(a) image

(b) solution

(BF/bal, $E_1 = 4097$, $t = 4.0$ sec, $r^+ = 0.71$, n sols = 34)

Figure C.18: Duck, 6 dof, Isolated Scene

C.11 Duck, 6 DOF, Cluttered

(BF/max, $E_1 = 4552$, $t = 5.5$ sec, $r^+ = 0.74$, n sols = 47)

Figure C.19: Duck, 6 dof, Cluttered Scene

(a) image

(b) solution

(BF/bal, $E_1 = 4071$, $t = 5.8$ sec, $r^+ = 0.76$, n sols = 90)

Figure C.20: Duck, 6 dof, Cluttered Scene

C.12 Duck, 6 DOF, Occluded

(a) image

(b) solution

(c) image, rotated view

(d) solution, rotated view

(BF/bal, $E_1 = 4105$, $t = 7.6$ sec, $r^+ = 0.72$, n sols = 184)

Figure C.21: Duck, 6 dof, Occluded Scene

Appendix D

Paper Reprints

Efficient and Reliable Template Set Matching for 3D Object Recognition

Michael Greenspan^{†‡}

Pierre Boulanger[‡]

[†]Department of Systems
and Computer Engineering
Carleton University, Ottawa, Canada

[‡]Visual Information Technology Group
Institute for Information Technology
National Research Council Canada

Abstract

Object recognition in range image data is formulated as template set matching. The object model is represented as a set of voxel templates, one for each possible pose. The set of all templates is composed into a binary decision tree. Each leaf node references a small number of templates. Each internal node references a single voxel, and has two branches, T and F. The subtree branching from the T branch contains the subset of templates which contain the node voxel. Conversely, the subtree branching from F branch contains the subset of templates which do not contain the node voxel. Traversing the tree at any image location executes a point probe strategy. It efficiently determines a good match with the template set by interrogating only those elements which discriminate between the remaining possible interpretations.

The method has been implemented for a number of different heuristic tree design and traversal methods. Results are presented of extensive tests for two objects under isolated, cluttered, and occluded scene conditions. It is shown that there exist traversal/design combinations which are both efficient and reliable, and that the method is robust.

1 Introduction

Feature extraction is a major paradigm in object recognition. The central concept is that visual scenes contain common low level primitives or features, such as lines and corners. Subsequent to the extraction of these features, recognition occurs within the space of the arrangement of features, rather than the much larger space of pixels. Feature extraction has, however, proven to be a difficult problem in its own right. The visual world is not a line drawing, and identifiable edges do not readily occur in intensity images. Further, the more robust feature extraction methods tend to be elaborate and therefore computationally expensive.

There are some object recognition methods that do not follow the feature extraction paradigm, such as the Interpretation Tree [1] and Geometric Hashing [2], both of which can operate directly at the raw data level. Other examples are the Hough Transform [3] and its close relative, simple template matching.

The main limitation of template matching is its expense. The brute-force exhaustive method is to translate the template to all possible image locations, and compare it with the overlaid image window. The comparison is done by calculating a similarity metric, such as the sum of absolute differences, or the sum of square differences.

Much of the literature on template matching has been concerned with devising ways to reduce this computational expense. Nagel and Rosenfeld [4] reduced the expense of matching a single template by ordering the sequence with which the template and image pixel pairs were compared. Those template pixels with a high expected difference from a randomly selected image pixel were matched first. This ordering increased the likelihood that an error threshold, which signified a mismatch at the location, was exceeded prior to comparing all pairs.

Ramapriyan [5] directly addressed the issue of efficiently matching a set of templates. The template set was organized into a tree structure, where the leafs corresponded to individual templates. The intermediate nodes corresponded to a single representative template (RT), which was the union of all descendant templates. Matching began at the root of the tree, and as the tree was traversed, the node whose RT best matched the image location was expanded further. Experimental results on synthetic images showed a performance improvement factor of 4 for a set of 36 templates. A significant difference between Ramapriyan's method and the work presented here is that, in our case, only a single template ele-

ment is referenced at each internal node, representing the intersection, rather than the union, of all descendant leaf templates. This not only improves efficiency, but also increases the size of the template set which can be encoded by such a scheme.

This paper continues in Section 2 with a description of the method, starting with a simple example in 2D. In Section 3, three heuristic methods of constructing the tree from the template set are presented. Issues of tree traversal are discussed in Section 4, and five traversal methods are presented, one of which is guaranteed to return the highest correlating template. In Section 5, a series of experiments are reported which evaluate the reliability and efficiency of various combinations of tree designs and traversals. The paper concludes in Section 6 with a summary and a discussion of future research directions.

2 Description of Method

2.1 Simple 2D Example

The basic approach can be best described with a simple 2D example. Assume that we wish to recognize the L-shaped object in the 2D binary image illustrated in Figure 1(a), where the black *on* pixels are the surface of an object, and the white *off* pixels are the image background. The object has a fixed scale with respect to the image, and can be translated anywhere in the image with four possible orientations (L, $\neg$, $\top$, and $\bot$).

Let us randomly select image pixel p_1 and hypothesize that it lies on the surface of the object. This hypothesis will be true if there is any translation of the four allowable orientations of the model which intersects p_1 . We define an *interpretation* of the hypothesis as a translation of a model template such that some template pixel uniquely intersects with p_1 . As each of the 4 templates has exactly 4 pixels which can be translated to intersect with p_1 , there are 16 possible interpretations of the initial hypothesis.

Let us assume that each interpretation occurs with the same likelihood, i.e. there is no preferred orientation or translation of the model. We can determine the likelihood that a neighboring pixel of p_1 is *on* as the ratio of the number of times an interpretation intersects with the neighboring pixel over the total number of interpretations. This value is interpreted as the likelihood that a pixel intersects with the surface of the model, given the truth of the initial hypothesis.

The set of all such likelihoods in the neighborhood of p_1 are illustrated in Figure 1(b). One possible strategy is to query the image at the next most likely *on* pixel. The result from this query will allow us to limit the number of possible interpretations. For example,

Figure 1: (a) 2D binary image, (b) likelihood of $p \in M$ in seed neighborhood

if we query the image at pixel p_2 and find it is *off*, then the hypothesis can only be true for interpretations which intersect p_1 and do not intersect p_2 . The set of possible interpretations remaining after such a query outcome are illustrated in Figure 2.

A new set of likelihoods is constructed from this remaining set of interpretations, and again the most likely surface pixel is queried. The process iterates in this manner until either a unique interpretation is found, thereby identifying and localizing the object, or all interpretations are ruled out as impossible, thereby refuting the hypothesis. If the hypothesis is refuted, then another image *on* pixel can be selected in place of p_1 , and the process repeats.

By using the knowledge of the object model to guide the sequence of image points to query, the object templates can be matched more efficiently than in a straightforward correlation. In the above example, a brute-force correlation of all templates would require between 4 and 16, and on average 10 pixel queries per image pixel. It can be shown that, following the proposed strategy, at most 5 queries per image seed are required. This difference in efficiency becomes more pronounced for the recognition of more complicated models.

2.2 Extension to 3D

The approach described in the above simple 2D example extends directly to the use of 3D range data. A set of templates is generated for each object model, with a single template representing a particular model pose. The model is rotated throughout its allowable rotation space, and at each pose, its surface is quantized into a voxel space. The resulting voxel space is called an *r-template*, as it has a purely rotational pose in the model-centric coordinate frame. Only those voxels which are non-occluded w.r.t. a specified sensor vantage are included in the r-template. Examples of 4

Figure 2: Remaining Interpretations

poses and the associated r-templates of a carved duck object are illustrated in Figure 3, where the sensor's line of sight is assumed to be rotated slightly up and to the right from that of the reader.

Each r-template is next translated so that some surface-valued voxel element lies at the origin. The complete template set therefore consists of all possible translations of all r-templates. In practise, only a small number of translations per r-template are used, to reduce the size of the template set and improve efficiency.

Each template represents a single view of the model in a specific pose. Given a continuous measurement space, there exist an infinite number of such poses. The quantization of the image into a discrete voxel space, however, imposes an equivalent quantization of pose space, which results in a finite-sized discrete pose space.

We tessellate pose space based upon an icosahedral viewing sphere, which is similar to Aspect Graph and EGI methods. Each vertex of the icosahedron is a viewing direction in model-centric coordinates, or a discrete model pose in world coordinates. One difference in our usage is that, whereas previous methods use a 2D viewing sphere, we allow rotations around the line of sight resulting in a 3D viewing sphere. Each icosahedron vertex therefore gives rise to a number of discrete pose coordinates, with the resolution of the rotation around the viewing direction commensurate with the angular separation between adjacent icosahedron vertices.

A binary decision tree is then composed from the template set, each leaf node of which references a small

Figure 3: Object Poses and Templates

number of templates. Each internal node references a single voxel, called the *node voxel*, and has a *true* (T) and *false* (F) branch. The subtree descending from the T branch contains that subset of templates which include the node voxel. Conversely, the subtree emanating from the F branch contains that subset of templates which do not include the node voxel.

Recognition proceeds by first voxelating the range image, and then randomly selecting an image *seed* voxel. As the tree is traversed all node voxels are considered relative to this seed. As each node is encountered, the value of its node voxel is queried in the image. If the node voxel is *surface-valued* (S -valued), then the T branch is followed. Otherwise, if the voxel is *free*, *occluded*, or *unknown*-valued, then the F branch is followed.

When a leaf node is encountered, a full correlation is calculated for the constituent leaf templates at the image seed location. If the correlation value is greater than a predefined threshold, then the template is declared as a potential solution, subject to further verification.

The process iterates by selecting each image voxel in turn as the seed.

2.3 Relation to Geometric Probing

Geometric probing is a minimalist technique whereby the object under investigation is probed in a structured adaptive sequence [6]. The method presented here has been previously described as a form of sequential hypothesis testing [7]. It can also be described as a geometric probing strategy, which are known to be representable as decision trees [8].

The probe function used here differs from other probe functions described in the literature [9]. In the finger probe function of Cole and Yap [6], a probe was defined to be a directed line l , and a probe outcome as the point p of contact between l and the boundary of a scene object. This probe function can be expressed Q_f which queries the scene with the specified probe

and returns the probe outcome ;

$$p = Q_f(l) \quad (1)$$

All other geometric probe functions can be cast in a similar expression.

We define a *discrete point probe* function, where the probe is a point p and the probe outcome is a discrete symbolic value k ;

$$k = Q_d(p) \quad (2)$$

In our case, there are only two possible outcomes to each probe ($k = T$ or F), making it a binary point probe function.

3 Tree Construction

The objective of decision tree construction is to generate a tree representation which has suitable characteristics for the problem at hand. The most common suitability criterion is minimization w.r.t. expected traversal cost. Another possibility is to reduce or minimize the rates of misclassification. Moret [10] has shown these (and other) criteria are in general *pairwise incompatible* which means that no tree representation can simultaneously optimize both.

The two known ways of generating optimal decision trees are through Dynamic Programming or Branch-and-Bound methods. It is also known that tree optimization is NP-complete. Given the large sizes of our template sets, it is unlikely that any optimal method would be efficient enough to be practical.

An alternative to optimal tree construction is to apply heuristics and generate a tree which is sub-optimal but nevertheless effective. Sub-optimal tree construction proceeds as a top-down process. Initially the complete set of templates is associated with the root node and is partitioned into subsets by selecting a node voxel. A new child node is then branched off of the root node for each template subset. The process recurses for each new node until the node template sets are small enough to terminate at leaf nodes.

The node template sets are partitioned at each step by selecting a discriminating voxel using a heuristic criteria, of which there are two main alternatives. The *balance* heuristic attempts to spawn equal-sized subtrees from each internal node. It does so by choosing that voxel which divides the node template sets into roughly equal sized sets. The *information* heuristic chooses that voxel which will, upon interrogation, yield the greatest amount of added information to the process.

Rather than restricting ourselves to one particular sub-optimal method, our approach has been to generate and compare the results of three heuristic criteria;

- **Balance:** For each node, a voxel is selected that is present in roughly half of the templates in the node template set. The selected node therefore partitions the set into two subsets of roughly equal size. If more than one voxel provides an equally good partition, then one of the alternatives is chosen randomly.
- **Max Information:** For a given node template set, a voxel is selected which is common to the maximum number of templates in the set.
- **Min Information:** This is the dual of the Max Information criterion. The voxel is selected that is common to the smallest non-empty subset of the node template set.

4 Tree Traversal

Tree traversal is the process of visiting a subset of the tree nodes in an ordered fashion, with the goal of efficiently determining a leaf template which matches the given image location. We determine the goodness of a match with an absolute sum of differences correlation function; any other standard correlation measure should also work equally well.

The simplest type of traversal is *nontentative depth-first*. Starting from the root node, the sequence of nodes visited passes from parent to child, terminating when a leaf node is encountered. This is the most efficient way of traversing a tree requiring only $\log n_L$ visits for n_L leaf nodes.

If the image data were perfect, then the nontentative depth-first traversal would be guaranteed to always produce the best solution, i.e. that template which correlates best for the given seed voxel. In practise, however, errors caused by noise, occlusions, and quantization effects cause some voxels in the image voxel map to have a value different from the ideal. The result is that any branching decision in this traversal may be incorrect, causing it to resolve to an incorrect solution.

The *reliability* of a traversal method is a measure of its ability to resolve to a correct, or near correct, solution, given that errors exist in the data. The reliability can be improved, and in some cases optimized, through the use of *backtracking* or *tentative* traversal. This reliability improvement comes at the expense of efficiency, as a larger portion of the tree nodes are visited.

4.1 Guaranteed Traversal

Branch-and-Bound methods are known to be optimal if, at each visited node, there exists a conservative bounding estimate of the quantity under optimization.

This is also known as A^* search. In our case, the correlation value is maximized, and the bounding estimate at node t_j is an upper estimate of the correlation of any leaf template descending from t_j . We denote the bounding estimate at t_j as $\hat{r}_j^+$, and determine its value as follows.

First, the tree is preprocessed so that each t_j encodes the size n_j^+ of the largest template associated with any of the leaf nodes descending from the subtree with t_j as its root. If t_j is itself a leaf node, then n_j^+ is trivially the size of its largest associated template.

As the tree is traversed, a running count $n_{\pi_j}^S$ is maintained along each partially expanded path π_j of the number of S-valued voxels encountered along π_j . A count $n_{\pi_j}^T$ is also maintained which indicates the number of nodes along π_j that are descended from the T branch of their parent node. For example, if all nodes along π_j have descended from T branches, then $n_{\pi_j}^T$ would be equal to the length of π_j . If a single node descended from a F branch, then $n_{\pi_j}^T$ would be one less than the path length, etc.

An expression for the conservative estimate $\hat{r}_j^+$ of the maximum correlation value of any template rooted at t_j is :

$$\hat{r}_j^+ = \frac{n_j^+ + n_{\pi_j}^S - n_{\pi_j}^T}{n_j^+} \quad (3)$$

During traversal, as each node t_j is encountered, $\hat{r}_j^+$ is calculated and compared to two values. The first is a threshold r_t which is the minimal acceptable correlation for a correct solution. The second is the current maximum correlation value r^+ for any leaf node which has been visited during the traversal. At any visited t_j , if $\hat{r}_j^+$ exceeds both r_t and r^+ , then the current path potentially leads to the best solution, and is expanded further. Otherwise, it is certain that any matches at leaf nodes descendant from t_j will necessarily correlate less than either r_t or r^+ , so π_j is pruned from further evaluation. Whenever a leaf node is encountered, a full correlation is evaluated for each of its associated templates, and the value of r^+ is updated.

Whenever a leaf node is encountered or a path is pruned, traversal continues by backtracking to the immediate parent node and evaluating the alternative path. This continues until all remaining paths have been either ruled out or expanded fully to the leaf node. The traversal is guaranteed to identify the solution with maximum correlation value, and at no additional cost returns all encountered solutions that exceed r_t .

The limitation of the Guaranteed traversal is that it is computationally expensive. In the worst case every node is visited, making it $O(n_L)$ for n_L leaf nodes, which is no better than (and when overheads are considered, somewhat worse than) a linear pass through the template set. In the best case, it can perform closer to $O(\log n_L)$. In our case, it was determined experimentally that only a small number of paths were pruned and that the efficiency was not much better than exhaustive matching. For this reason, Guaranteed traversal was not further considered in the experimentation.

4.2 Heuristic Traversal

With the aim of improving efficiency, four types of heuristic traversal were implemented.

4.2.1 Best-First

Best-First traversal is a simple non-tentative depth-first method. At each node, the node voxel is queried and, if found to be S-valued, the T branch is followed. Otherwise, the F branch is followed. This traversal has complexity $O(\log n_L)$.

4.2.2 Beam

The Beam traversal employs a limited form of backtracking. Rather than making the decision to follow a branch based just on the direct descendants of each node, the Beam traversal looks ahead into the tree to a greater depth. From each visited node, all paths of depth d are further expanded, and an estimate of the maximum correlation is calculated. The direct descendant node is followed whose depth d subtree results in the largest correlation estimate.

The decision of which child to follow is based upon more information than the Best-First traversal, so the Beam traversal tends to be both more reliable and less efficient. For each path node, all subtrees of depth d are expanded, resulting in a complexity of $O(2^d \log n_L)$.

4.2.3 N-Error

The N-Error traversal was motivated by error coding methods in information theory. A path is considered as a string of the binary symbols T and F , indicating which branch was followed from the parent nodes. Any symbol along the path can be incorrect, and we assume that each symbol has the same probability of being in error.

Initially a Best-First traversal is invoked, which results in a path from the root to some leaf node. It is

then assumed that any branching decision could have been in error. For each node in the path, the traversal then backtracks and takes the alternate branch, which generates a new path resolving at a different leaf node.

We assume that there are e erroneous symbols in any path string, and expand all alternatives. Given an average path length of $\log n_L$, the number of possible variations from with e errors is :

$$P \binom{\log n_L}{e} = \frac{\log n_L!}{(\log n_L - e)!} \quad (4)$$

The complexity of this traversal is therefore $O(\log^e n_L)$.

4.2.4 Heuristic Branch-and-Bound

This traversal follows the same branch-and-bound regime as Guaranteed traversal. It is based on the assumption that any branching errors encountered along a traversed path will be randomly distributed along that path in a uniform way. For any partial path π_j of length n_π , we estimate the correlation $\hat{r}_j$ of a template associated with a descendant leaf node to be the ratio of the number of encountered to ideal S-valued voxels :

$$\hat{r}_j = n_{\pi_j}^S / n_{\pi_j}^T, \quad (5)$$

Unlike the $\hat{r}_j^+$ of the Guaranteed traversal, $\hat{r}_j$ may fall on either side of the resultant measured correlation value. It therefore cannot be used to drive an optimal traversal but may provide a closer estimate of the true correlation value, particularly when n_π is still small.

Although this traversal has the same $O(n_L)$ complexity as Guaranteed traversal, it will on average perform more efficiently as it uses a more liberal pruning criterion.

5 Experimentation

The objective of the experimentation was to determine whether there exists combinations of tree designs and traversals which were both efficient and reliable. The method was implemented and tested on two objects, a toy boat and a carved duck. Models were constructed for each object by first acquiring a covering set of range images at a number of different aspects (~ 10). These range images were then combined into a single registered point cloud, from which a triangular mesh polyhedron (*t-mesh*) was generated.

The tree representations were constructed directly from each t-mesh. In order to reduce the time to generate the trees, the space of object orientations was restricted so that only the top half of the object was considered. Two rotational dimensions of the general

object	design	# nodes	traversal cost		
			min	max	E_T
boat	balance	7927	11	60	14.3
	max info	8737	2	61	15.7
	min info	31155	2	15578	7790.
duck	balance	8027	11	21	13.0
	max info	12487	10	49	17.0
	min info	22179	2	11090	5546.

Table 1: Tree Metrics (5760 templates/tree)

six dimensional pose space were therefore only half-dimensions. For this pose space, there were 5760 templates generated per object. Trees were constructed for each object using all 3 heuristic criteria. Some metric properties of the trees are listed in Table 1.

Range images were acquired with an autosynchronous scanner [11] and a Biris sensor mounted on a linear motion platform. Images were acquired of each object under a variety of isolated and cluttered scene conditions. In the isolated scenes, only the object of interest was present, except possibly for a planar background. In the cluttered scenes, there were a number of different objects arbitrarily placed in the image.

Twenty images were acquired for each object in each scene condition. The object was randomly positioned within its pose space for each image, and an attempt was made to position the object approximately uniformly throughout its pose space for each image set.

For each image, recognition was attempted for every combination of the 4 heuristic traversals and the 3 tree designs. Examples of some positive results are illustrated in Figures 13 and 14. The figure captions report the tree traversal/design combination, the E_1 efficiency metric (defined in Section 5.1.1), the time t to traverse the tree, the correlation value r^+ of the illustrated solution, and the number of solutions returned whose correlations values exceeded r_t . Note that the tests were executed on a single MIPS R8000 processor (which was not particularly powerful).

5.1 Performance Metrics

The performance of each tree traversal/design combination was evaluated based upon one efficiency and two reliability metrics.

5.1.1 Efficiency

Let a tree containing n nodes encode t templates. Let a traversal of the tree for all s image seeds result in c correlation calculations. The efficiency E_1 of a given traversal is defined as the ratio of the number of correlation calculations required for exhaustive template

Figure 4: Isolated Boat Trials

Figure 6: Cluttered Boat Trials

Figure 5: Isolated Boat Trials, Low Range

Figure 7: Cluttered Boat Trials, Low Range

matching with that number actually executed during traversal ;

$$E_1 = \frac{t \times s}{c} \quad (6)$$

5.1.2 Reliability

We measured reliability in two ways, the first of which was by manually inspecting the solutions. A match was judged to be correct if it was close enough to the object's pose that an iterative closest point method [12] would drive it to a true solution. We denote this metric as R_1 . For a single recognition event, R_1 can be either 1 (indicating success) or 0 (indicating failure);

$$R_1 = \begin{cases} 1 & \text{if success} \\ 0 & \text{if failure} \end{cases} \quad (7)$$

The second reliability metric is denoted R_2 and was determined automatically by examining the maximum correlation value r^+ of all solutions. If r^+ was greater than some threshold r_t then the recognition was deemed successful, and R_2 was set to r^+ . Otherwise, the recognition event had failed, and R_2 was set to 0.

$$R_2 = \begin{cases} r^+ & \text{if } r^+ \geq r_t \\ 0 & \text{if } r^+ < r_t \end{cases} \quad (8)$$

Of the two metrics, R_1 is the most faithful, but is not automatic. The effectiveness of R_2 depends largely

on the value of r_t . If the value of r_t is too high, then the chance of rejecting a correct solution is greater, and the false negative error rate α will increase. A large r_t will also reduce the chance of accepting a poor solution so the false positive error rate β will simultaneously decrease. Conversely, if r_t is too low, then α will decrease, and β will increase.

5.2 Efficiency vs. Reliability

The efficiency vs. reliability results from these trials were plotted in Figures 4 to 11, where the diamond indicates R_1 and the cross R_2 . We were particularly interested in assessing whether there existed combinations of tree traversal and design which exhibited moderately high efficiency and reliability ratings. Some of the significant plot points are annotated according to their tree traversal/design method as follows: BF=Best-First, BM=beam, NE=N-Error, HB=Heuristic Branch-and-Bound, bal=Balance, min=Min-Information, max=Max-Information.

These plots indicate that there was a tradeoff between efficiency and reliability, and that some combinations of tree traversal/design exhibited both high reliability and high efficiency. Examples are the BF/bal and BF/max combinations, which fared well in both the Isolated and Cluttered trials. The plots also show that the R_2 metric was closely related to R_1 , which supports the belief that a threshold can be selected to

Figure 8: Isolated Duck Trials

Figure 10: Cluttered Duck Trials

Figure 9: Isolated Duck Trials, Low Range

Figure 11: Cluttered Duck Trials, Low Range

automatically determine successful recognition events.

For the same object, if we compare the results of the Isolated and Cluttered trials (Figures 4 and 6, 5 and 7, 8 and 10, and 9 and 11), it can be seen that the traversal/design combinations fall in basically the same order across these plots. This indicates that the method behaves similarly in both isolated and cluttered scenes.

The most striking observation comes from comparing the results from the same scene conditions across different objects (Figures 4 and 8, 5 and 9, 6 and 10, and 7 and 11). It can be seen that the different combinations of tree traversal/design behave similarly across the two objects. As there was no particular similarity between the shapes of these two objects, this result indicates that the good behaviour of these tree traversal/design combinations may generalize to any object.

One final observation that bears discussion is the exceptionally good performance of the BF/min combination in the isolated boat trials (Figure 4). The efficiency rating was over 2.5×10^4 while the reliability was a respectable $\bar{R}_1 = 0.6$. Comparing this result to the expected traversal costs tabulated in Table 1, it can be seen that this result is at odds with the theory, which predicts that the min info tree should behave less efficiently than the other designs. One possible

reason for this discrepancy is that the calculated values of E_T hold only when the input signal is an instance of one of the leaf classes, which is less often the case in a cluttered scene. Note that this result did not hold in the cluttered boat trials (Figure 6) or any of the duck object trials.

5.3 Robustness

An experiment was designed to investigate the degradation in reliability as the noise level increased. We assumed that at the voxel level of detail the effect of noise was the assignment of erroneous voxel values, with the number and distribution of voxels affected dependant upon the level of noise and the noise model. We assumed that the noise was in a random uniform

Figure 12: Noise vs. Reliability ($\bar{R}_1$)

(a) image

(b) solution

(BF/max, $E_1 = 4552$, $t = 5.5$ sec, $r^+ = 0.74$, n sols = 47)

Figure 13: Duck, Cluttered Scene

(a) image

(b) solution

(c) image, rotated view

(d) solution, rotated view

(BF/max, $E_1 = 4395$, $t = 7.2$ sec, $r^+ = 0.74$, n sols = 330)

Figure 14: Boat, Occluded Scene

distribution, and randomly reassigned a specified percentage of the voxels to a particular noise value.

The method was tested for 10 images of the boat under 4 degrees of positional freedom (i.e. a table-top scenario) in isolated scene conditions. The Balanced tree design was used in combination with four traversal methods (Best-First, N-Error, Beam, and Heuristic Branch-and-Bound). The $\bar{R}_1$ reliability metric was plotted as a function of % noise in Figure 12. Note that a significant amount of image noise would be required to change the voxel space by 10% at the selected voxel resolution of $\rho = 0.015\text{m}$.

The results indicated that reliability degraded gracefully as noise level increased, and we concluded that the method was robust to noise. As expected, the more extensive traversals were more immune to the effects of noise. The traversals ranked in order of increasing robustness were : Best-First, N-Error, Beam, and Heuristic Branch-and-Bound. This is the same ordinal ranking with respect to increasing computational expense.

The effects of partial occlusion are similar to those of noise in that some fraction of the interrogated voxels will have non-ideal values. Some trials were run in which the object of interest was partially occluded, and an example of a positive result is illustrated in Figure 14. While anecdotal, these tests supported the claim that the method is robust to occlusions.

6 Conclusions

We draw the following three conclusion from these experiments;

- There exist combinations of tree traversal and design method which are both efficient and reliable. In some cases, the efficiency was as much as 4 orders of magnitude greater than that of brute-force template matching.
- The performance of the traversal/design combinations is relatively constant for the various scene conditions accross the two objects tested. As there was no particular similarity between the shapes of the boat and duck objects, this result implies that these performances may be true in general for any object shape, or for a large class of object shapes.
- The method degrades gracefully with increasing noise, and is therefore robust.

In future work we plan to evaluate the performance of the method for a larger object database. One issue is to determine the tradeoffs between having separate

trees, as opposed to encoding all objects into one large tree. The current work used a minimalistic scheme, in which a single point was interrogated at each tree node. It is further planned to evaluate the performance as additional information is considered at each node. The method will also be implemented on a parallel computational platform.

References

- [1] W.E.L. Grimson and T. Lozano-Perez. Model-based object recognition and localization from sparse range or tactile data. *Intl. Jour. Rob. Res.*, 3(3):3-35, Fall 1984.
- [2] Y. Lamdan and H.J. Wolfson. Geometric hashing : A general and efficient model-based recognition scheme. In *2nd International Conference on Computer Vision*, pages 238-249, December 1988.
- [3] R. Krishnapuram and D. Casasent. Determination of three-dimensional object location and orientation from range images. *IEEE Trans. PAMI*, 11(11):1158-1167, November 1989.
- [4] Roger N. Nagel and Azriel Rosenfeld. Ordered search techniques in template matching. *Proceedings of the IEEE*, 60:242-244, February 1972.
- [5] H. K. Ramapriyan. A multilevel approach to sequential detection of pictorial features. *IEEE Transactions on Computers*, 1:66-78, January 1976.
- [6] Richard Cole and Chee K. Yap. Shape from probing. *Journal of Algorithms*, 8:19-38, 1987.
- [7] Michael Greenspan. The sample tree : A sequential hypothesis testing approach to 3d object recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 772-779, 1998.
- [8] Esther M. Arkin, Joseph S. B. Mitchell, Henk Meijer, David Rappaport, and Steven S. Skiena. Decision trees for geometric models. *International Journal of Computational Geometry and Applications*, 8(3):343-363, 1998.
- [9] Steven S. Skiena. Interactive reconstruction via geometric probing. *Proceedings of the IEEE*, 80(9):1364-1383, September 1992.
- [10] Bernard M.E. Moret. Decision trees and diagrams. *ACM Comp. Surv.*, 14(4):593-623, December 1982.
- [11] Marc Rioux. Laser range finder based on synchronized scanners. In Thierry Bosch and Marc Lescure, editors, *Analysis and Interpretation of Range Images*. Springer-Verlag, 1990. Also, *Appl. Opt.* 23(21): 3837-3844; 1984, NRC 24528.
- [12] Paul J. Besl and Heil D. McKay. A method for registration of 3d shapes. *IEEE Trans. PAMI*, 14(2):239-256, February 1992.

The Sample Tree : A Sequential Hypothesis Testing Approach to 3D Object Recognition

Michael Greenspan

Visual Information Technology
Institute for Information Technology
National Research Council of Canada

Department of Systems
and Computer Engineering
Carleton University

M50 Montreal Rd., Ottawa, Ontario, Canada, K1A 0R6

Michael.Greenspan@nrc.ca

Abstract

A method is presented for efficient and reliable object recognition within noisy, cluttered, and occluded range images. The method is based on a strategy which hypothesizes the intersection of the object with some selected image point, and searches for additional surface data at locations relative to that point. At each increment, the image is queried for the existence of surface data at a specific spatial location, and the set of possible object poses is further restricted. Eventually, either the object is identified and localized, or the initial hypothesis is refuted.

The strategy is implemented in the discrete domain as a binary decision tree classifier. The tree leaf nodes represent individual voxel templates of the model. The internal tree nodes represent the union of the templates of their descendant leaf nodes. The union of all leaf node templates is the complete template set of the model over its discrete pose space. Each internal node also references a single voxel which is the most common element of its child node templates. Traversing the tree is equivalent to efficiently matching the large set of templates at a selected image seed location.

The process is approximately 3 orders of magnitude more efficient than brute-force template matching. Experimental results are presented in which objects are reliably recognized and localized in 6 dimensions in less than 60 seconds within noisy and significantly occluded range images.

1 Introduction

The tree structure is among the most well-utilized representations in object recognition, and is particularly common in Correspondence Space methods where the tree nodes represent bindings between model and data features. The tree provides an efficient means of organizing and searching the exponen-

tially large space, and follows either a *data-based* or a *model-based* search strategy. In the data-based approach, the nodes are associated with scene features, and are paired with model features during the matching process. The complete data-based tree is rarely explicitly represented, but rather is partially generated as a by-product of the search process. This is because tree construction is costly, and the node features are only known once the image has been acquired and the features extracted. The data-based tree can also be very space expensive, particularly when the scene is cluttered with many unknown objects.¹

Conversely, in the model-based approach, the nodes are associated with model features which are known independent of any specific image. The tree can therefore be constructed offline, and its structure optimized to improve recognition performance.

Whether based upon a tree representation or not, the tendency in object recognition is toward the use of high level image features. The Interpretation Tree method, for example, was originally based on sparse low level point data in images of isolated objects [2]. It was subsequently deemed necessary to include edge and plane features to deal with the increase in the size of the search space that resulted from more general images with clutter and occlusions [3]. The trend is also evident in Geometric Hashing [4, 5] and other methods. Some recent recognition methods which operate directly on low level data and do not require a feature extraction preprocessing phase include [6, 7].

The use of high level features effectively reduces the space of possibilities in the matching phase, and lends structure to the search process. A reliance on

¹Chen and Kak [1] argue that, in the case of single object scenes, the search space of the data-based tree will be smaller than that of the model-based tree.

method	elements	tree type
IT	low[2] & hi[3]	data
Bonsai [9]	hi	data
3D-POLY [1]	hi	data
LFF [10]	hi	model
CAGD [11]	hi	model
Strategy Tree [12]	hi	model
Sample Tree	low	model

Table 1: Classification of Tree Methods

feature extraction, however, can be problematic. The robustness of extracted features may depend upon extrinsic factors, such as lighting conditions in 2D intensity images, and sensor vantage in 3D range images [8]. Feature extraction can also be computationally expensive, and is often more so than the subsequent matching. Methods based upon a particular feature set are restricted in application to objects that contain some of those features.

Table 1 categorizes some of the major tree-based object recognition methods as either model- or data-based, and using either low or hi level scene elements. We consider a low level element to be a raw image point, and a hi level element to be any feature that requires extraction preprocessing, such as an edge, plane, quadric surface, point of maximum curvature, etc.

Our object recognition method is based upon a strategy which incrementally queries the image at specific point locations for the existence of surface data. At each increment, the set of possible object poses which can satisfy the queried image points is reduced, until only a small number of potential matches remain. The approach uses low level point data and a model-based tree structure, called a *sample tree*. The sample tree is a binary decision tree classifier and is constructed offline automatically from the model geometry. The method uses image data directly and, as there is no reliance upon feature extraction preprocessing, can handle free-form surfaces.

The remainder of this paper is organized as follows: in Section 2, 3D object recognition is defined as a constraint satisfaction problem. Section 3 describes our approach as a sequential hypothesis testing process. Section 4 describes the implementation of the approach into a complete object recognition system, and Section 5 presents some experimental results. The paper concludes in Section 6 with a summary and a discussion of further research.

2 Recognition as Constraint Satisfaction

Let a scene S consist of a set of surfaces, and let a range image I of S be defined as a set of n points $\{p_i\}_1^n$ which samples the surfaces in S . The points are described in sensor frame coordinates, and in keeping with most range data acquisition technologies, we assume that any $p_i \in I$ correspond only to portions of the surfaces in S which are non-occluded with respect to the sensor vantage.

Let M be a surface model of a rigid object, which may be transformed homogeneously within S to any 6 dimensional pose $\tilde{\theta}$. We denote an instance of M with pose $\tilde{\theta}$ as $M|_{\tilde{\theta}}$. The objective of object recognition is to determine all occurrences of $M|_{\tilde{\theta}}$ in the subspace of S that is imaged by I .

Let $\Gamma(M|_{\tilde{\theta}}, S)$ be the set of surfaces defined by the projection of the non-occluded surfaces of $M|_{\tilde{\theta}}$ onto the closest corresponding surfaces of S . Thus, for every non-occluded point on the surface of the model in the given pose, there will exist a corresponding closest projected point in the scene;

$$p_m \in M|_{\tilde{\theta}} \implies \exists p_g \in \Gamma(M|_{\tilde{\theta}}, S) \quad (1)$$

$$\text{s.t. } p_g \in S \text{ and } |p_m - p_g| \leq |p_m - p_s| \forall p_s \in S$$

One way to measure the goodness of a match of $M|_{\tilde{\theta}}$ in S is to consider the magnitude of the area of $\Gamma(M|_{\tilde{\theta}}, S)$, and the magnitude of the volume bound by $M|_{\tilde{\theta}}$ and $\Gamma(M|_{\tilde{\theta}}, S)$. In the case of a perfect match, for example, the area of $\Gamma(M|_{\tilde{\theta}}, S)$ would equal that of $M|_{\tilde{\theta}}$, and the volume between them would have a magnitude of zero.

Let $A(S_i)$ denote the magnitude of the area of a set of surfaces S_i and let $V(S_i, S_j)$ denote the magnitude of the minimal volume bound by two surface sets. Object recognition can then be defined as a process of finding all values of $\tilde{\theta}$ which satisfy the following constraints:

$$|A(M|_{\tilde{\theta}}) - A(\Gamma(M|_{\tilde{\theta}}, S))| < \epsilon_A \quad (2)$$

$$V(M|_{\tilde{\theta}}, \Gamma(M|_{\tilde{\theta}}, S)) < \epsilon_V \quad (3)$$

where ϵ_A and ϵ_V are thresholds determined by the model geometry, the accuracies of the sensor and model, and the noise and occlusion properties of the image and scene.

These constraints form a necessary but not sufficient condition for recognition. There may be arbitrary scene surfaces which satisfy the constraints, but

are not part of the object of interest. There may also be other objects which are similar in shape to the object of interest, and satisfy the constraints in certain poses. Approximations of these constraints are the basis of ICP algorithms which are commonly used in pose refinement processes [13].

3 Sequential Hypothesis Testing

Our objective is to find all values of $\tilde{\theta}$ such that $M|_{\tilde{\theta}}$ satisfies the constraint inequations. The exhaustive method of simply transforming the model through all values of $\tilde{\theta}$ and estimating the constraint function values is not feasible, as the general 6 dimensional pose space is large. Rather, the space can be searched more efficiently with a sequential, or multi-stage, hypothesis test.

Let Θ be the 6D pose space of M in S , and let Θ_i be a subspace of Θ . Thus, $M|_{\Theta_i}$ denotes the set of all instances of M with some pose $\tilde{\theta}$ in Θ_i ;

$$M|_{\Theta_i} = \bigcup_{\forall \tilde{\theta} \in \Theta_i} M|_{\tilde{\theta}} \quad (4)$$

The sequential hypothesis test commences with the random selection of a range data point p_0 in I , called the *seed point*. We formulate an initial hypothesis that the seed point lies on the surface of the model ;

$$H_0 : p_0 \in M \quad (5)$$

Note that H_0 makes no assumptions about the pose of M . Indeed, at this stage the hypothesis does not establish correspondence between p_0 and any particular surface point of M . For H_0 to be true, p_0 can be any point on the surface of M . The truth of H_0 therefore restricts the possible poses of M to the subset Θ_0 which contains only those $\tilde{\theta}$ such that $M|_{\tilde{\theta}}$ intersects with p_0 .

At the next stage, the hypothesis is extended by querying I at another point p_1 in $M|_{\Theta_0}$. If surface data exists in I at p_1 (denoted $p_1 \in I$) then we form the hypothesis which asserts that an instance of M exists in some pose such that both p_1 and p_0 intersect its surface. Alternately, if no surface data exists at p_1 , then we hypothesize that p_0 falls on the model surface and p_1 does not:

$$H_1 = \begin{cases} p_1 \in M, p_0 \in M & \text{if } p_1 \in I \\ p_1 \notin M, p_0 \in M & \text{if } p_1 \notin I \end{cases} \quad (6)$$

We respectively call these the *TRUE* and *FALSE* alternatives, and denote them by H^T and H^F . We continue extending the hypothesis by querying I at specific points relative to the initial seed point. At any

stage i , one of the two alternatives is extended:

$$H_i = \begin{cases} H_i^T : p_i \in M, H_{i-1} & \text{if } p_i \in I \\ H_i^F : p_i \notin M, H_{i-1} & \text{if } p_i \notin I \end{cases} \quad (7)$$

At each successive stage, the cardinality of the pose subspace is reduced until eventually only a small number of model poses can satisfy the hypothesis, which are subsequently verified.

3.1 Sample Tree

The above described sequential hypothesis testing procedure can be naturally represented by a binary decision tree classifier. In general, classification is a pattern recognition process where an input vector in some n -dimensional feature space is determined to fall within some feature space region, or *class*. In cases where either the dimension of the input vector or the number of classes is large, efficiency can be improved with a multi-stage classifier, which considers only a small number of feature vector components at each stage, thereby reducing the set of possible classes that need be compared at each successive stage.

The root of a decision tree represents the entire set of classes, and each interior node represents a subset of the classes associated with its parent node. A leaf node represents either a single class, or a small set of classes. In addition to a class set, each non-terminal node has an associated feature set and decision rule. When traversing the tree, the decision rule is applied at each node to the associated feature subset of the input vector to determine which descendant branch to follow. The decision rule simultaneously measures the similarity of the feature subset with all remaining possible classes, and reduces their number accordingly. Decision trees have been used successfully in medical image processing [14] and OCR [15, 16] applications.

In the sample tree each node represents a subset of model poses $M|_{\Theta_i}$ (the *node set*), and a relative point location (the *node point*). The root node set contains a reference to every model pose, i.e. $M|_{\Theta}$. Descendant nodes partition their parent node set into two disjoint node sets, which are discriminated by the presence or absence of the parent node point. The node sets of any descendant node is the subset of model poses which best matches the node points encountered along the path to that node. A leaf node contains a small (possibly empty) node set, and no node point.

The structure of a decision tree is primarily dependent upon which variable (in our case, which relative point) is interrogated at each tree node. Much of the literature on decision tree classifiers is concerned with methods of constructing trees which optimize certain

properties, the most common objective being to minimize the expected traversal cost. For even a modest number of classes, there are a huge number of possible tree structures, so it is infeasible to simply generate and compare all alternatives. There are construction methods which are known to minimize the expected testing cost, such as dynamic programming. All truly optimal methods, however, are computationally expensive, and have indeed been shown to be NP-complete [17].

Moret [18] has identified two classes of heuristic that are used in sub-optimal decision tree generation. In the *information heuristic*, a node variable is chosen that maximizes the amount of additional information that is gained at that stage along the path traversal. In the *splitting heuristic*, a variable is chosen which divides the remaining class set into 2 disjoint sets which are effectively discriminated by the node variable.

We have employed a splitting heuristic which, at any node along the traversed path, samples the point in space which is most likely to fall on the greatest number of object poses in the node set. This is similar to the heuristic used by Moret based upon variable activity. The effect of this rule is that the difference between the cardinality of the two child node sets will be maximal, and the tree will be very unbalanced. This is contrary to most tree construction methods, which attempt to minimize the expected traversal cost by balancing the tree structure. The rationale for our heuristic is that in all but the simplest images, the randomly selected seed point p_0 is likely not to fall on the object of interest. At each extension, we therefore select a point which falls on the surface of the greatest number of $M|_{\tilde{\theta}}$ in the current node set. If as expected p_0 does not fall on the model surface, then the cardinality of the remaining $M|_{\Theta_1}$ will be small, as will be the number of subsequent hypothesis extensions required to refute H_0 . In summary, our tree is designed to be efficient at refuting H_0 , which will be the correct outcome in the majority of cases.

4 Implementation

The sample tree forms the basis of the 4 phase recognition system illustrated in Figure 1. Rather than operating in continuous space, the first 2 phases of the process map the problem into a coarse resolution discrete domain, with $\mathbf{I}$ represented as a voxel map. Each image voxel can take one of 4 values: **surface (S)** if the voxel intersects some image surface data, **free (F)** if the voxel lies between the sensor head and a surface-valued voxel, **occluded (O)** if the voxel lies behind a surface-valued voxel w.r.t. the sensor head, or **unknown (U)** if the voxel lies outside

Figure 1: Process Diagram

of the sensing frustum. An example of the surface-valued voxels of a voxelated range image is illustrated in Figure 2. The resolution ρ of this voxel map is set at 0.015 m , which is the resolution used in all tests reported in Section 5. It can be seen that this resolution is coarser than is typically considered acceptable in image processing.

Operating in the discrete domain offers two advantages. First, it provides an efficient means to index a particular point in the image space to determine the existence of surface-valued data, an operation which is used extensively in the first two phases of the method. Second, the quantization of 3D space into voxels imposes an equivalent quantization on the pose space Θ . For any given voxel resolution, there is a continuous neighborhood Θ_1 which map all $M|_{\tilde{\theta}} \in M|_{\Theta_1}$ into identical voxel templates. The discrete pose space requires at most one representative pose value from each neighborhood, which effectively enumerates and reduces the space of possible model poses.

When mapped into the discrete domain, traversing the sample tree is equivalent to an efficient template matching process, as described in [19]. This is similar to the *template tree* representation of Ramapriyan [20], an important difference being that the intermediate

Figure 2: Voxelated Range Image, $\rho = 0.015 \text{ m}$

nodes of the template tree represent the union of all descendant leaf node templates, whereas the sample tree intermediate nodes represent their intersection. Each query of a sample tree node voxel is equivalent to subtemplate matching, with the subtemplate consisting of a single element.

4.1 Phase 1 : Hypothesis Generation

The hypothesis generation phase consists of traversing the sample tree. The traversal proceeds by first selecting a surface-valued image voxel as the (discrete) seed point. The location of this seed voxel is the base which is used to relate all node locations to the voxel map frame. As each tree node is visited, the image voxel which is offset from the seed voxel by the node voxel location is queried. If this voxel is surface-valued, then the *TRUE* path is followed, which is equivalent to expanding H^T . Otherwise, H^F is expanded and the traversal follows the *FALSE* path.

Assume that we have generated a path of k nodes from the tree root to a leaf node by following the above described depth-first traversal. If the selected seed point did indeed lie on the object's surface, and if the image data were perfect, then the simple depth-first search would be the most efficient means of reaching the correct leaf node. In practice, image noise and occlusions and voxel map quantization errors can cause any image voxel queried along the path to have an erroneous value. This would cause the non-tentative traversal to follow an incorrect branch, necessarily excluding the correct pose.

We improve the robustness of the traversal by assuming that the value of any one of the voxels queried between the root and the leaf could have been incorrect. Once the leaf node is reached, the traversal backtracks to each path node in turn and evaluates the alternative branch. As there are $k - 2$ intermediate nodes between the root and the leaf, a single back-

tracking traversal follows $k - 1$ paths and generates $k - 1$ leaf solutions.

The traversal is repeated for all possible image seeds. Whenever a leaf node is reached, the model pose values contained therein are stored, along with the seed voxel location, for use in the subsequent phase.

For a sample tree with n leaf nodes, the average path length k is approximately equal to $\log n$, where n is dependent upon the relationship between the size of the object and the quantization coarseness². For each seed voxel, $k - 1$ paths are traversed, resulting in at most $k(k - 1) \simeq \log^2 n$ image voxel queries. The time complexity of the sample tree traversal for s image seeds and n leaf nodes is therefore $O(s \log^2 n)$.

4.2 Phase 2 : Template Verification

The intent of phase 1 is to be fairly liberal in generating hypotheses. As a result, some shallow paths may hypothesize poses for which only a small number of points have been tested. The purpose of phase 2 is to efficiently remove all phase 1 results which are unlikely to satisfy the constraints. This is accomplished by template matching. All poses resulting from the first phase are verified at the voxel resolution by template correlation. Only those model poses whose correlation exceeds a specified threshold are passed to the subsequent phase. The threshold must be restrictive enough to filter out the majority of incorrect poses, but forgiving enough to be robust to noise, quantization errors, and occlusions. In the results reported in Section 5, the correlation threshold was set at 80%.

²For example, assume an object of surface area A voxels has a discrete pose space cardinality h . If b distinct templates resolve to each leaf node, then $n = hA/b$.

4.3 Phase 3 : Closest Point Verification

In many cases, the highest ranked phase 2 solutions correctly identify the object. Sometimes however, due to the coarseness of the quantization, an incorrect solution may have a higher correlation measure than a correct solution. This will occur less frequently with a finer discrete resolution, but too fine a resolution negates the benefits of quantization.

In phase 3, the phase 2 results are ranked by estimating the values of the constraint functions at the image data resolution. In preprocessing, two point sets are generated by sampling the surfaces of M at a coarse and fine resolution. These are called the *coarse* and *fine surface samples*. The closest point metric is calculated by transforming the coarse surface sample in turn to the pose value of each phase 2 solution. For each point in the transformed surface sample, the identity of and separation distance to the closest image point is determined. The area constraint (equation 2) is estimated by counting the number of image points that are within a specified threshold distance, which is approximately equal to the voxel resolution. The volume constraint (equation 3) is estimated by summing the separation distances.

The closest point metric and constraint values are then estimated more accurately using the fine surface sample for the top ranked solutions from the coarse surface sample estimate.

4.4 Phase 4 : ICP Refinement

The phase 3 solutions are likely to have a pose error commensurate with the quantization resolution. To refine the pose estimate, an ICP process is applied to the highest ranked phase 3 solutions. The constraints are estimated again following the pose refinement to further verify the match.

5 Experimental Results

The method has been implemented and tested on two objects, a toy boat and a hand-carved duck. The duck was chosen specifically as it was a free-form surface with no planes, edges, or other obvious features. Executing on a standard sequential computer (an SGI workstation with a MIPS R8000 CPU) the results compare favorably to other known methods, with recognition often achieved in less than 60 seconds.

For each model, the sample trees and supporting structures were generated in the offline preprocessing stage. The tree structures were quite large, and their top-down construction was computationally expensive, requiring a number of days of processing for each tree. Some characteristics of the trees used in the described tests are listed in Table 2. The item *brute-force ops.* refers to the number of image voxel

object	Boat	Duck
# of nodes	184383	115375
av. depth	21	21
max. depth	78	49
templates/leaf	5	5
brute-force ops.	43.1×10^6	13.9×10^6

Table 2: Tree Characteristics

image	Boat	Duck A	Duck B
<i>t phase 1 (sec)</i>	9.8	6.5	3.3
<i>t phase 2</i>	26.6	10.9	9.5
<i>t phase 3</i>	0.5	47.4	9.9
<i>t phase 4</i>	5.6	1.9	2.4
<i>t total</i>	42.5	66.7	25.1
# seeds	1141	901	803
# phase 1 sols	70364	56153	45941
# phase 2 sols	6	965	212
<i>E</i>	8433	5373	5671
<i>r (%)</i>	90	82	82
<i>phase 3 cp (mm)</i>	16	10	7
<i>phase 4 cp (mm)</i>	6	9	6

Table 3: Test Results

queries that would be required to perform an exhaustive brute-force template correlation at a single seed location over the complete set of templates referenced in the tree leafs.

The objects were positioned arbitrarily in scenes of various degrees of clutter and occlusion. In some cases, the objects were partially occluded by other objects, or partially outside of the sensor's field of view. Dense images were acquired using a laser range scanner based on the autosynchronous scanning method [21]. The scanner, which was mounted on the end effector of a robotic manipulator, was oriented at various angles and distances to the table top environment. The particular range scanner used was an early prototype which had fallen into disuse, and so the images were noisier than is achievable with current sensor versions.

Examples of some successful trials are illustrated in Figures 3 to 5. For each trial the range image is illustrated from the sensor's vantage, and two views are shown of the object superimposed onto the image at the recognized pose. The timing results of the various phases of these trials are tabulated in Table 3. Also included are: the number of solutions resultant from each phase; the phase 1 efficiency *E* (the ratio of the

Figure 3: Boat

Figure 4: Duck A

Figure 5: Duck B

number of brute-force template element comparisons and the number of comparisons resulting from the tree traversal); the best phase 2 correlation measure r , and the average phase 3 and 4 closest point separations.

6 Summary

A novel object recognition method has been presented, and early implementation results are very promising. Some of the benefits of this method are;

- **efficiency:** The method is fast. In experiments, the full 6 dimensional pose of objects were determined in a few tens of seconds. The time complexity of the tree traversal phase is related to the base 2 logarithm of the number of templates encoded in the tree.
- **free-form surfaces:** As there is no dependence upon a feature extraction preprocessing phase, the method is applicable to any shape of rigid object, including free-form surfaces.
- **fully automatic:** The input to the offline processing is either a CAD model or a cloud of points which sample the surface of the object. The tree generation is fully automatic, requiring no user interaction other than the selection of a handful of parameters at the offset.
- **scalable:** The tree may be composed of a template set derived from a number of objects. Increasing the number of objects has a less than linear performance penalty on the tree traversal phase. Practical limitations to the number of objects simultaneously encoded into a single tree are likely to be the storage requirements and the expense of generating the tree.

- **parallelizable:** The process can be executed independently across seeds (seed parallelism) or piped across phases (phase parallelism).
- **other sensing modes:** The method can be applied directly to 2D binary images (as in OCR) or volumetric data (as in [22]).

The continuing research will focus on revisiting some of the heuristics and algorithms used with the goal of improving upon the overall reliability and efficiency of the method. It may be beneficial to add a hashing phase following the hypothesis generation phase, as in [7]. Additional tree construction and traversal heuristics will be investigated, and efficiency and reliability metrics will be thoroughly evaluated.

Acknowledgments

I would like to thank my thesis supervisors, Pierre Boulanger and Bernie Pagurek, for their support and many helpful suggestions. Thanks also to Gerhard Roth for the use of his t-mesh models and Guy Godin for his ICP routine.

References

- [1] C.H. Chen and A.C. Kak. A robot vision system for recognizing 3-d objects in low-order polynomial time. *IEEE Trans. SMC*, 19(6):1535–1563, Nov/Dec 1989.
- [2] W.E.L. Grimson and T. Lozano-Perez. Model-based object recognition and localization from sparse range or tactile data. *Intl. Jour. Rob. Res.*, 3(3):3–35, Fall 1984.
- [3] W.E.L. Grimson and T. Lozano-Perez. Localizing overlapping parts by searching the interpretation tree. *IEEE Trans. PAMI*, PAMI-9(4):469–482, July 1987.
- [4] Y. Lamdan and H.J. Wolfson. Geometric hashing : A general and efficient model-based recognition scheme. In *2nd International Conference on Computer Vision*, pages 238–249, December 1988.
- [5] Y. Lamdan, J.T. Shwartz, and H.J. Wolfson. Object recognition by affine invariant matching. In *Proc. IEEE Conf. Comp. Vis. Pat. Rec.*, pages 335–344, June 1988.
- [6] Andrew Edie Johnson and Martial Hebert. Surface registration by matching oriented points. In *IM97 : Proceedings of the International Conference on Recent Advances in 3-D Digital Imaging and Modeling*, pages 121–128, May 1997.
- [7] Chin Seng Chua and Ray Jarvis. Point signatures: A new representation for 3d object recognition. *Intl. Jour. Comp. Vis.*, 25(1):63–85, 1997.
- [8] Frank P. Ferrie, Shailendra Mathur, and Gilbert Soucy. Feature extraction for 3-d model building and object recognition. In Anil K. Jain and Patrick J. Flynn, editors, *Three-Dimensional Object Recognition Systems*, pages 57–88. Elsevier, 1993.
- [9] P.J. Flynn and A.K. Jain. Bonsai: 3-d object recognition using constrained search. *IEEE Trans. PAMI*, 13(10):1066–1075, October 1991.
- [10] R.C. Bolles and P. Horaud. 3dpo: A three-dimensional part orientation system. *Intl. Jour. Rob. Res.*, 5(3):3–26, Fall 1986.
- [11] C. Hansen and T.C. Henderson. Cagd-based computer vision. *IEEE Trans. PAMI*, 11(11):1181–1193, November 1989.
- [12] Katsushi Ikeichi. Generating an interpretation tree from a cad model for 3d-object recognition in bin-picking tasks. *Intl. Jour. Comp. Vis.*, 1:145–165, 1987.
- [13] Paul J. Besl and Heil D. McKay. A method for registration of 3d shapes. *IEEE Trans. PAMI*, 14(2):239–256, February 1992.
- [14] R.J.B. Giesen, A.L. Huynen, R.G. Aarnink, J.J.M.C.H. de la Rosette, F.M.J. Debruyne, and H. Wijkstra. Construction and application of hierarchical decision tree for classification of ultrasonic prostate images. *Medical & Biological Engineering & Computing*, pages 105–109, March 1996.
- [15] Q.R. Wang and C.Y. Suen. Analysis and design of a decision tree based on entropy reduction and its application to large character set recognition. *IEEE Trans. PAMI*, PAMI-6(4):406–417, July 1984.
- [16] Q.R. Wang and C.Y. Suen. Large tree classifier with heuristic search and global training. *IEEE Trans. PAMI*, PAMI-9(1):91–102, January 1987.
- [17] L. Hyafil and R.L. Rivest. Constructing optimal binary decision trees is np-complete. *Comm. of the ACM*, 5:15–17, May 1976.
- [18] Bernard M.E. Moret. Decision trees and diagrams. *ACM Comp. Surv.*, 14(4):593–623, December 1982.
- [19] Michael A. Greenspan. A decision tree classifier for object recognition in range imagery. In *Proceedings of Vision Interface 98*, Vancouver, B.C., June 1998.
- [20] H. K. Ramapriyan. A multilevel approach to sequential detection of pictorial features. *IEEE Transactions on Computers*, 1:66–78, January 1976.
- [21] Marc Rioux. Laser range finder based on synchronized scanners. In Thierry Bosch and Marc Lescure, editors, *Analysis and Interpretation of Range Images*. Springer-Verlag, 1990. Also, *Appl. Opt.* 23(21): 3837–3844; 1984, NRC 24528.
- [22] Gill Barequet and Micha Sharir. Partial surface and volume matching in three dimensions. *IEEE Trans. PAMI*, 19(9):929–948, 1997.

A Decision Tree Classifier for Object Recognition in Range Imagery

Michael Greenspan

Visual Information Technology
Institute for Information Technology
National Research Council of Canada

Department of Systems
and Computer Engineering
Carleton University

M50 Montreal Rd., Ottawa, Ontario, Canada, K1A 0R6

Michael.Greenspan@nrc.ca

Abstract

A method is presented for efficient and reliable object recognition within noisy, cluttered, and occluded range images. The method is based on efficiently matching a set of voxel templates generated by rotating a model exhaustively through its pose space. The template set is composed into a binary decision tree classifier, with the tree leaf nodes representing individual voxel templates of the model. The internal tree nodes represent the union of the templates of their descendant leaf nodes, and the union of all leaf node templates is the complete template set of the model oriented over its discrete pose space. Each internal node also references a single voxel which is common to its child node templates.

Traversing the tree is equivalent to efficiently matching the large set of templates at a selected image location. Experiments have shown the process to be approximately 4 orders of magnitude more efficient than brute-force template matching. Results are presented in which recognition is achieved in less than 20 seconds within noisy and significantly occluded range images.

1 Introduction

Most object recognition methods involve a feature extraction preprocessing phase, followed by a matching phase which relates the extracted image features to those in a model database. The use of high level features effectively reduces the space of possibilities in the matching phase, and lends structure to the matching process. A reliance on feature extraction can, however, be problematic. The robustness of extracted features may depend upon

extrinsic factors, such as lighting conditions in 2D intensity images, and sensor vantage in 3D range images [1]. Feature extraction can also be computationally expensive, and is often more so than the subsequent matching. Further, methods based on a particular feature set are restricted to the recognition of objects that contain some features from that set.

An alternative to feature extraction is template matching. An object to be recognized is represented as a template, which is a subimage described in low level image elements. Matching is performed by sliding the template over all possible image locations, and calculating a correlation metric.

In its straightforward implementation, template matching can be inefficient. For m template pixels and n image pixels, $m^2(n - m + 1)^2$ operations are required to score the template at all image locations. Rosenfeld et al. [2, 3] reduce this expense by sequencing the order in which the elements are compared, thereby increasing the likelihood that an error threshold signifying a mismatch is exceeded prior to comparing all pairs. Other proposed methods to improve efficiency include multistage subtemplate matching [4], multiresolution [5] and hierarchical structures [6, 7].

While the above cited methods focus on improving the efficiency of matching a single template against an image, other methods improve the efficiency of matching a set of templates. The best known of these is the Hough Transform [8] which is equivalent to matching a set of templates from a given class of objects [9]. Ramapriyan [10] organized a set of templates into a tree structure with leaf nodes corresponding to individual templates. At each intermediate node, the input template was

compared with a representative template comprised of the union of all descendant templates, and the value of the score determined which branch to follow. Experimental results on 2D images with a set of 36 templates showed an improvement of a factor of 4 over the brute-force method.

Our approach is based upon compiling a set of 3D templates, generated by rotating a model exhaustively through its pose space, into a binary decision tree classifier. A decision tree classifier is a pattern recognition structure which efficiently associates an input signal with one of a possibly large number of classes. In our case, the leaf nodes represent small sets of templates, and the intermediate nodes represent the union of all descendant sets. As the tree is traversed, a single template element, which is the intersection of all descendant node templates, is used to differentiate between the two children nodes. We have found this construct to be both reliable and many orders of magnitude more efficient than the brute-force approach.

The complete method is organized into 4 processing phases. In the first hypothesis generation phase, the tree is traversed for each (discrete) image point, and likely template matches (i.e. model poses) are identified at each location. The two subsequent verification phases rank the results first in the discrete and then the continuous domain. A final refinement phase improves the estimated pose of the match using a standard Iterative Closest Point (ICP) method [11]. A diagram of the main offline and online processing phases, and their associated data structures, is illustrated in Figure 1.

The remainder of this paper is organized as follows: in Section 2, 3D object recognition is defined as a constraint satisfaction problem, and the quantization of the problem domain is justified. Section 3 describes the hypothesis generation phase and addresses aspects of the decision tree construction and traversal. Section 4 describes the verification phases. Experimental results are presented in Section 5, and the paper concludes in Section 6 with a summary and a discussion of some future research issues.

2 Recognition as Constraint Satisfaction

Let a scene S consist of a set of surfaces, and let a range image I of S be defined as a set of n points $\{p_i\}_1^n$ which samples the surfaces in S . The points are described in sensor frame coordinates, and in keeping with most range data acquisition technolo-

Figure 1: Process Diagram

gies, we assume that any $p_i \in I$ correspond only to portions of the surfaces in S which are non-occluded with respect to the sensor vantage.

Let M be a surface model of a rigid object, which may be transformed homogeneously within S to any 6 dimensional pose $\tilde{\theta}$. We denote an instance of M with pose $\tilde{\theta}$ as $M|_{\tilde{\theta}}$. The objective of object recognition is to determine all occurrences of $M|_{\tilde{\theta}}$ in the subspace of S that is imaged by I .

Let $\Gamma(M|_{\tilde{\theta}}, S)$ be the set of surfaces defined by the projection of the non-occluded surfaces of $M|_{\tilde{\theta}}$ onto the closest corresponding surfaces of S . Thus, for every non-occluded point on the surface of the model in the given pose, there will exist a corresponding closest projected point in the scene;

$$p_m \in M|_{\tilde{\theta}} \implies \exists p_g \in \Gamma(M|_{\tilde{\theta}}, S) \quad (1)$$

$$\text{s.t. } p_g \in S \text{ and } |p_m - p_g| \leq |p_m - p_s| \forall p_s \in S$$

One way to measure the goodness of a match of $M|_{\tilde{\theta}}$ in S is to consider the magnitude of the area of $\Gamma(M|_{\tilde{\theta}}, S)$, and the magnitude of the volume bound by $M|_{\tilde{\theta}}$ and $\Gamma(M|_{\tilde{\theta}}, S)$. In the case of a perfect match, for example, the area of $\Gamma(M|_{\tilde{\theta}}, S)$ would

equal that of $M|_{\vec{\theta}}$, and the volume between them would have a magnitude of zero.

Let $\mathcal{A}(S_i)$ denote the magnitude of the area of a set of surfaces S_i and let $\mathcal{V}(S_i, S_j)$ denote the magnitude of the minimal volume bound by two surface sets. Object recognition can then be defined as a process of finding all values of $\vec{\theta}$ which satisfy the following constraints:

$$|\mathcal{A}(M|_{\vec{\theta}}) - \mathcal{A}(\Gamma(M|_{\vec{\theta}}, S))| < \varepsilon_A \quad (2)$$

$$\mathcal{V}(M|_{\vec{\theta}}, \Gamma(M|_{\vec{\theta}}, S)) < \varepsilon_V \quad (3)$$

where ε_A and ε_V are thresholds determined by the model geometry, the accuracies of the sensor and model, and the noise and occlusion properties of the image and scene.

These constraints form a necessary but not sufficient condition for recognition. There may be arbitrary scene surfaces which satisfy the constraints, but are not part of the object of interest. There may also be other objects which are similar in shape to the object of interest, and satisfy the constraints in certain poses. Approximations of these constraints are the basis of ICP algorithms which are commonly used in pose refinement processes [11].

The exhaustive method of simply transforming the model through all values of $\vec{\theta}$ and estimating the constraint function values is not feasible, as the general 6 dimensional pose space is large. Rather, we must rely upon heuristic methods of searching this space.

2.1 Domain Quantization

Rather than operating in continuous space, the first two phases of the process map the problem into a coarse resolution discrete domain, with I represented as a voxel map, and each $M|_{\vec{\theta}}$ as a set of templates of the voxels that intersect with the surface of the model in the specified pose. The quantization of range images into voxels, and the related *octree* representation, is a technique that has been used to advantage in mesh generation [12, 13], image registration [14], gaze planning [15], and recently object recognition [16]. While many recognition methods quantize some mapping of the image signal [17, 18], to the author's knowledge, the only known method in which voxel template matching has been used directly is [19].

Each voxel in the range image voxel map takes on one of four values :

- **surface (S)** : the voxel intersects with the surface of the model ;

a) range image, sensor view

b) range image, rotated view

c) image voxel map

Figure 2: Voxlated Range Image

- **free (F)** : the voxel lies between the sensor head and a surface-valued voxel ;
- **occluded (O)** : the voxel lies behind a surface-valued voxel from the sensor head ;
- **unknown (U)** : the voxel lies outside of the sensing frustum.

An example of the surface-valued voxels of a voxelated range image is illustrated in Figure 2. It can be seen that the voxel resolution used in the experiments is coarser than is typically considered acceptable in image processing.

Mapping the problem into the discrete domain offers two advantages. First, it provides an efficient means to index a particular point in space to determine the presence or absence of surface-valued range

data, an operation which is used extensively in the first two phases of the method. Second, the quantization of 3D space into voxels imposes an equivalent quantization of the pose space : for any given voxel resolution, there is a continuous neighborhood of poses which map the model surfaces into identical voxel templates. The discrete pose space requires at most one representative pose value from each neighborhood, which effectively enumerates and reduces the space of possible model poses.

For example, in the experiments reported in Section 5, the voxel resolution is chosen to be 15 mm^3 , which is ~ 100 times coarser than the raw range data. The discrete pose space is based upon $1/2$ of a level-2 icosahedron with a solid angle of ~ 10 degrees between rotational coordinates, resulting in 5760 unique rotational pose coordinates.

3 Hypothesis Generation

3.1 RT-Template Definition

In the hypothesis generation phase, for each surface-valued image voxel, a set of candidate model poses is generated which are likely to satisfy the constraint inequations (2) and (3). For each pure rotational value $\vec{\theta}$ of the discrete pose space, we define a *rotation template (r-template)* as the set of non-occluded voxels which intersect with $M|_{\vec{\theta}}$. Similarly, a *rotation translation template (rt-template)* is defined as an r-template that has been translated so that one of its member voxels lies at the origin. For a given pose space quantization, the number of r-templates of a specific model will be equal to the number of unique rotation pose space coordinates, and the number of rt-templates will be equal to the number of r-templates multiplied by the average r-template size.

3.2 Template Matching As Decision Tree Classification

The objective of the Hypothesis Generation Phase is to find candidate matches of every rt-template against every image voxel. Due to the large number of rt-templates, the brute-force method of simply translating each rt-template to each surface-valued image voxel is too costly, even at a reduced resolution.

To improve efficiency, we compose the rt-template set into a binary decision tree classifier. In general, classification is a pattern recognition process whereby an input vector is determined to fall within

some feature space region, or *class*. In cases where either the dimension of the input vector or the number of classes is large, efficiency can be improved with a multi-stage classifier, which considers only a small number of feature vector components at each stage, thereby reducing the set of possible classes that need be compared at each successive stage.

The root of a decision tree represents the entire set of classes, and each interior node represents a subset of the classes associated with its parent node. A leaf node represents either a single class, or a small set of classes. In addition to a class set, each non-terminal node has an associated feature set and decision rule. When traversing the tree, the decision rule is applied at each node to the associated feature subset of the input vector to determine which descendant branch to follow. The decision rule simultaneously measures the similarity of the feature subset with all remaining possible classes, and reduces their number accordingly. Decision trees have been used successfully in medical image processing [20] and OCR [21, 22] applications.

We call our decision tree a *sample tree*, because it is used to schedule the sampling of an image. Each node of the sample tree represents a set of rt-templates (the node set), and a relative voxel location (the node voxel). The root node set contains a reference to every rt-template. Descendant nodes partition their parent node set into two disjoint node sets, which are discriminated by the presence or absence of the parent node voxel. The node sets of all descendant nodes is the subset of rt-templates which best matches the node voxels encountered along the path to that node. A leaf node contains a small (possibly empty) node set, and no node voxel.

For example, for parent node P with node set N_p and relative voxel location V_p , all of the rt-templates referenced in the TRUE child node set will contain V_p , and all of those referenced in the FALSE child node set will not.

3.3 Sample Tree Construction

The structure of a decision tree is primarily dependant upon which variable (in our case, which relative voxel) is interrogated at each tree node. Much of the literature on decision tree classifiers is concerned with methods of constructing trees which optimize certain properties, the most common objective being to minimize the expected traversal cost.

For even a modest number of classes, there are a huge number of possible tree structures, so it is infeasible to simply generate and compare all alter-

natives. There are construction methods which are known to minimize the expected testing cost, such as dynamic programming. All truly optimal methods, however, are computationally expensive, and have indeed been shown to be NP-complete [23].

Moret [24] has identified two classes of heuristic that are used in sub-optimal decision tree generation. In the *information heuristic*, a node variable is chosen that maximizes the amount of additional information that is gained at that stage along the path traversal. In the *splitting heuristic*, a variable is chosen which divides the remaining class set into 2 disjoint sets which are effectively discriminated by the node variable.

We have employed a splitting heuristic which, at any node along the traversed path, samples the point in space which is most likely to fall on the object. When constructing a tree node, this splitting rule selects the voxel which belongs to the largest number of rt-templates in the node set. This is similar to the heuristic used by Moret based upon variable activity.

The effect of this rule is that the difference between the cardinality of the two child node sets will be maximal, and the tree will be very unbalanced. This is contrary to most tree construction methods, which attempt to minimize the expected traversal cost by balancing the tree structure. A rational for favouring an unbalanced structure is presented in the following section.

3.4 Tree Traversal

Tree traversal proceeds by first selecting a surface-valued image voxel, called the *seed voxel*. The location of the seed voxel is the base which is used to relate all tree voxels to the image voxel map frame.

If the image data were perfect, then a simple depth-first traversal would arrive at the correct leaf node most efficiently. As each node is visited, the image voxel which is offset from the seed voxel by the node voxel location is queried. If this voxel is surface-valued data, then the traversal follows the *TRUE* path. If not, then the traversal follows the *FALSE* path.

In practise, image noise, occlusions, and quantization errors can cause a non-tentative search to follow an incorrect path. We therefore allow some limited backtracking to follow other likely paths, at a slight efficiency cost. The termination condition is satisfied when either a predefined number of solutions has been generated, or a predefined number of leafs has been visited.

The traversal is repeated for all possible image

seeds. Whenever a leaf node is reached, the rt-templates contained in the leaf node set are stored, along with the seed voxel location, for use in the subsequent phase.

One way to interpret the tree traversal is as a form of scheduled subtemplate matching, where at each node the subtemplate consists of a single voxel. Another interpretation is that each tree traversal hypothesizes that the current seed voxel lies on the object, and then sequentially interrogates the image at locations relative to this seed to either support or refute this hypothesis.

This interpretation lends the following rational for the splitting rule used in tree construction: in all except the simplest cases, the object of interest will not monopolize the sensor's field of view, and the majority of image voxels will not fall on the object. If the initial seed point does not fall on the object, then as the path is followed, subsequent queries will likely encounter nonsurface-valued voxels. Those nodes with the smallest sets will then be followed, resulting in quick termination at a leaf. The splitting rule therefore results in an unbalanced tree structure so that traversal will terminate most efficiently in the majority of cases.

4 Verification

4.1 Template Matching Metric

In phase 2, for every phase 1 solution we perform a complete template match at the image voxel resolution. Each rt-template is translated so that it is aligned with its associated seed voxel. A simple correlation metric r is calculated by taking the ratio of the number of coincident surface voxels and the number of total rt-template voxels. If this measure exceeds a threshold value, then the result is passed to phase 3.

4.2 Closest Point Metric

In many cases, the highest ranked phase 2 solutions are correct. Sometimes however, due to the coarseness of the quantization, an incorrect solution may have a higher correlation measure than a correct solution. This will occur less frequently with a finer discrete resolution, but too fine a resolution will negate the benefits of quantization.

In phase 3, the phase 2 results are ranked by estimating the values of the constraint functions at the image data resolution. In preprocessing, two point sets are generated by sampling the surfaces of M at a coarse and fine resolution. These are called

the *coarse* and *fine surface samples*. The constraint function values are estimated by transforming the coarse surface sample in turn to the pose value of each phase 2 solution. For each point in the transformed surface sample, the identity of and separation distance to the closest image point is calculated. The area of overlap is estimated by counting the number of image points that are within a specified threshold distance, which is approximately equal to the voxel resolution. The volume of overlap is estimated by averaging the separation distances.

The closest point metric is subsequently estimated more accurately using the fine surface sample for the top 10 ranked solutions from the coarse surface sample estimate.

5 Experimental Results

The method has been implemented and tested on a hand-carved duck object, which was chosen specifically as it was free-form surface with no planes, edges, or other obvious features. Executing on a standard sequential computer (an SGI workstation with a MIPS R8000 CPU) the results compare favourably to other known methods, with recognition often achieved in less than 20 seconds.

The object was positioned arbitrarily in scenes of various degrees of clutter and occlusion. In some cases, the object was partially occluded by other objects, or partially outside of the sensor's field of view. Dense images were acquired using a laser range scanner based on the autosynchronous scanning method [25]. The scanner, which was mounted on the end effector of a robotic manipulator, was oriented at various angles and distances from the table top environment. The particular range scanner used was an early prototype which had fallen into disuse, and so the images were noisier than is achievable with current sensor versions.

Examples of three successful trials are illustrated in Figures 3 to 5. For each trial the range image is illustrated from the sensor's vantage, and two views are shown of the object superimposed onto the image at the recognized pose. The timing results of the various phases of these trials are tabulated in Table 1. Also included are: the number of solutions resultant from each phase; the phase 1 efficiency E (the ratio of the number of brute-force template element comparisons and the number of comparisons resulting from the tree traversal); the best phase 2 correlation measure r (as defined in Section 4); and an estimate of the volume constraint $\hat{V}()$ (the average closest point separation).

6 Summary

A novel object recognition method has been presented, and early implementation results are very promising. Some of the benefits of this method are;

- **efficiency:** The method is fast. In experiments, the full 6 dimensional pose of objects were determined in ≈ 20 seconds. The time complexity of the tree traversal phase is related to the base 2 logarithm of the number of templates encoded in the tree.
- **free-form surfaces:** As there is no dependence upon a feature extraction preprocessing phase, the method is applicable to any shape of rigid object, including free-form surfaces.
- **fully automatic:** The input to the offline processing is either a CAD model or a cloud of points which sample the surface of the object, and the tree generation is fully automatic, requiring no user interaction other than the selection of a handful of parameters at the offset.
- **scalable:** The tree may be composed of a template set derived from a number of objects. Increasing the number of objects has a less than linear performance penalty on the tree traversal phase. A practical limitation to the number of objects simultaneously encoded into a single tree is likely to be the storage requirements, as the size of the tree increases exponentially with the number of templates.
- **parallelizable:** The process can be executed independently across seeds (seed parallelism) or piped across phases (phase parallelism).
- **other sensing modes:** The method can be applied directly to 2D binary images (as in OCR) or volumetric data (as in [16]).

The continuing research will focus on revisiting some of the heuristics and algorithms used with the goal of improving upon the overall reliability and efficiency of the method. It may be beneficial to add a hashing phase following the hypothesis generation phase, as in [18]. Additional tree construction and traversal heuristics will be investigated, and efficiency and reliability metrics will be thoroughly evaluated.

Acknowledgements

I would like to thank my thesis supervisors, Pierre Boulanger and Bernie Pagurek, for their support

Figure 3: Image 1 : $t = 10.5$ seconds

Figure 4: Image 2 : $t = 14.0$ seconds

Figure 5: Image 3 : $t = 21.9$ seconds

<i>image</i>	<i>time (seconds)</i>				<i># of solutions</i>		E	r	$\dot{V}()$
	<i>total</i>	<i>phase 1</i>	<i>phase 2</i>	<i>phase 3</i>	<i>phase 1</i>	<i>phase 2</i>			
1	10.5	3.8	3.5	3.2	12660	78	17685	0.83	$5.3e-5$
2	14.0	3.8	3.9	6.3	14253	156	18022	0.94	$1.6e-5$
3	21.9	3.4	3.7	14.8	14024	335	17996	0.82	$1.2e-4$

Table 1: Test Results

and many helpful suggestions. Thanks also to Gerhard Roth for the use of his t-mesh models and Guy Godin for his ICP routine.

References

- [1] Frank P. Ferrie, Shailendra Mathur, and Gilbert Soucy. Feature extraction for 3-d model building and object recognition. In Anil K. Jain and Patrick J. Flynn, editors, *Three-Dimensional Object Recognition Systems*, pages 57–88. Elsevier, 1993.
- [2] Roger N. Nagel and Azriel Rosenfeld. Ordered search techniques in template matching. *Proceedings of the IEEE*, 60:242–244, February 1972.
- [3] Avraham Margalit and Azriel Rosenfeld. Using probabilistic domain knowledge to reduce the expected computational cost of template matching. *CGVIP*, 51:219–233, 1990.
- [4] Gordon J. Vanderbrug and Azriel Rosenfeld. Two-stage template matching. *IEEE Transactions on Computers*, C-26(4):384–393, April 1977.
- [5] Azriel Rosenfeld and Gordon J. Vanderbrug. Coarse-fine template matching. *IEEE Trans. SMC*, SMC-7:104–107, February 1977.
- [6] Robert Y. Wong and Ernest L. Hall. Sequential hierarchical scene matching. *IEEE Transactions on Computers*, C-27(4):359–366, April 1978.
- [7] Steven L. Tanimoto. Template matching in pyramids. *Computer Graphics and Image Processing*, 16:356–369, 1981.
- [8] J. Illingworth and J. Kittler. A survey of the hough transform. *CGVIP*, 44:87–116, 1988.
- [9] G. Stockman and A.K. Agrawala. Equivalence of hough curve detection to template matching. *Comm. of the ACM*, 20(11):820–822, November 1977.
- [10] H. K. Ramapriyan. A multilevel approach to sequential detection of pictorial features. *IEEE Transactions on Computers*, 1:66–78, January 1976.
- [11] Paul J. Besl and Heil D. McKay. A method for registration of 3d shapes. *IEEE Trans. PAMI*, 14(2):239–256, February 1992.
- [12] W.E. Lorenen and H.E. Cline. Marching cubes : A high resolution 3d surface reconstruction algorithm. In *Computer Graphics : Siggraph '87 Conference Proceedings*, volume 21, pages 163–169, July 1987.
- [13] Gerhard Roth and Eko Wibowoo. An efficient volumetric method for building closed triangular meshoes from 3-d image and point data. In *Graphics Interface '97*, pages 173–180, May 1997.
- [14] Régis Houde, Jacques Tremblay, Frédéric Auclair, and Denis Laurendeau. Pose determination using octrees : A comparative survey. In *Vision Interface 95*, pages 220–227, 1995.
- [15] Dimitri Papadopoulos-Orfanos and Francis Schmitt. Automatic 3-d digitization using a laser rangefinder with a small filed of view. In *3DIM97 : Proceedings of the International Conference on Recent Advances in 3-D Digital Imaging and Modeling*, pages 60–67, Ottawa, Ontario, Canada, May 1997.
- [16] Gill Barequet and Micha Sharir. Partial surface and volume matching in three dimensions. *IEEE Trans. PAMI*, 19(9):929–948, 1997.
- [17] Andrew Edie Johnson and Martial Hebert. Surface registration by matching oriented points. In *3DIM97 : Proceedings of the International Conference on Recent Advances in 3-D Digital Imaging and Modeling*, pages 121–128, May 1997.
- [18] Chin Seng Chua and Ray Jarvis. Point signatures: A new representation for 3d object recognition. *Intl. Jour. Comp. Vis.*, 25(1):63–85, 1997.
- [19] Timothy S. Newman and Anil K. Jain. Bidirectional template-matching for 3d cad-based inspection. *Proceedings of the SPIE*, 2183:257–265, 1994.
- [20] R.J.B. Giesen, A.L. Huynen, R.G. Aarnink, J.J.M.C.H. de la Rosette, F.M.J. Debruyne, and H. Wijkstra. Construction and application of hierarchical decision tree for classification of ultrasonic prostate images. *Medical & Biological Engineering & Computing*, pages 105–109, March 1996.
- [21] Q.R. Wang and C.Y. Suen. Analysis and design of a decision tree based on entropy reduction and its application to large character set recognition. *IEEE Trans. PAMI*, PAMI-6(4):406–417, July 1984.
- [22] Q.R. Wang and C.Y. Suen. Large tree classifier with heuristic search and global training. *IEEE Trans. PAMI*, PAMI-9(1):91–102, January 1987.
- [23] L. Hyafil and R.L. Rivest. Constructing optimal binary decision trees is np-complete. *Comm. of the ACM*, 5:15–17, May 1976.
- [24] Bernard M.E. Moret. Decision trees and diagrams. *ACM Comp. Surv.*, 14(4):593–623, December 1982.
- [25] Marc Rioux. Laser range finder based on synchronized scanners. In Thierry Bosch and Marc Les-cure, editors, *Analysis and Interpretation of Range Images*. Springer-Verlag, 1990. Also, *Appl. Opt.* 23(21): 3837–3844; 1984, NRC 24528.

Bibliography

- [1] David Marr. On the purpose of low-level vision. Memo 324, Massachusetts Institute of Technology Artificial Intelligence Laboratory, Dec. 1974.
- [2] Yolande Muller and Roger Mohr. Planes and quadrics detection using hough transform. In *Proceedings of the Seventh International Conference on Pattern Recognition*, pages 1101–1103, 1984.
- [3] R. Krishnapuram and D. Casasent. Determination of three-dimensional object location and orientation from range images. *IEEE Trans. PAMI*, 11(11):1158–1167, November 1989.
- [4] W.E.L. Grimson and T. Lozano-Perez. Model-based object recognition and localization from sparse range or tactile data. *Intl. Jour. Rob. Res.*, 3(3):3–35, Fall 1984.
- [5] Y. Lamdan, J.T. Schwartz, and H.J. Wolfson. On recognition of 3-d objects from 2-d images. In *Proc. IEEE Int. Conf. Rob. Aut.*, pages 1407–1413, April 1988.
- [6] Y. Lamdan, J.T. Schwartz, and H.J. Wolfson. Object recognition by affine invariant matching. In *Proc. IEEE Conf. Comp. Vis. Pat. Rec.*, pages 335–344, June 1988.
- [7] Y. Lamdan and H.J. Wolfson. Geometric hashing : A general and efficient model-based recognition scheme. In *2nd International Conference on Computer Vision*, pages 238–249, December 1988.
- [8] Thimas C. Henderson and Wo So Fai. The 3-d hough shape transform. *Pat. Rec. Ltrs.*, 2(4):235–238, 1984.

- [9] W.E.L. Grimson and T. Lozano-Perez. Localizing overlapping parts by searching the interpretation tree. *IEEE Trans. PAMI*, PAMI-9(4):469–482, July 1987.
- [10] Frank C.D. Tsai. Geometric hashing with line features. *Pat. Rec.*, 27(3):337–389, 1994.
- [11] Richard Cole and Chee K. Yap. Shape from probing. *Journal of Algorithms*, 8:19–38, 1987.
- [12] Steven S. Skiena. Interactive reconstruction via geometric probing. *Proceedings of the IEEE*, 80(9):1364–1383, September 1992.
- [13] Esther M. Arkin, Joseph S. B. Mitchell, Henk Meijer, David Rapaport, and Steven S. Skiena. Decision trees for geometric models. *International Journal of Computational Geometry and Applications*, 8(3):343–363, 1998.
- [14] D. Dobkin, H. Edelsbrunner, and C.K. Yap. Probing convex polytopes. In *Proceedings of the Eighteenth Annual ACM Symposium*, pages 424–432, 1986.
- [15] Michael Greenspan. The sample tree : A sequential hypothesis testing approach to 3d object recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 772–779, 1998.
- [16] Michael Greenspan. A decision tree classifier for object recognition in range imagery. In *Vision Interface 98*, pages 181–188, 1998.
- [17] Michael Greenspan and Pierre Boulanger. Efficient and reliable template set matching for 3d object recognition. In submitted to *3DIM99*, 1999.
- [18] Herbert J. Bernstein. Determining the shape of a convex n -sided polygon by using $2n+k$ tactile probes. *Information Processing Letters*, 22:255–260, April 1986.
- [19] Eugene Joseph and Steven Sol Skiena. Model-based probing strategies for convex polygons. *Computational Geometry: Theory and Applications*, 2:209–221, 1992.

- [20] Kelly A. Lyons and David Rappaport. An efficient algorithm for identifying objects using robot probes. *Visual Computer*, 10:452–458, 1992.
- [21] P.D. Alevizos, J.D. Boissonnat, and M. Yvinec. An optimal $o(n \log n)$ algorithm for contour reconstruction from rays. In *Proceedings of the Third Annual Symposium on Computational Geometry*, pages 162–170, 1987.
- [22] P.D. Alevizos, J.D. Boissonnat, and M. Yvinec. On the order induced by a set of rays: Application to the probing of non convex polygons. In *Proc. IEEE Int. Conf. Rob. Aut.*, pages 202–207, 1989.
- [23] J.D. Boissonnat and M. Yvinec. Probing a scene of non convex polyhedra. In *Proceedings of the Fifth Annual Symposium on Computational Geometry*, pages 237–246, 1989.
- [24] E.M. Arkin, M.T. Goodrich, J.S.B. Mitchell, D. Mount, C.D. Piatko, and S.S. Skiena. Point probe decision trees for geometric concept classes. In *Proceedings of the 3rd Workshop on Algorithms and Data Structures*, pages 95–106, 1993.
- [25] L. Hyafil and R.L. Rivest. Constructing optimal binary decision trees is np-complete. *Comm. of the ACM*, 5:15–17, May 1976.
- [26] Michael Lindenbaum and Alfred Bruckstein. Parallel strategies for geometric probing. *Journal of Algorithms*, 13:320–349, 1992.
- [27] R.E. Ellis, Edward M. Riseman, and Allen R. Hanson. Tactile recognition by probing: Identifying a polygon on a plane. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 632–637, 1986.
- [28] Patrice Belleville and Thomas C. Shermer. Probing polygons minimally is hard. *Computational Geometry: Theory and Applications*, 2:255–265, 1993.
- [29] Y. Shirai and M. Suwa. Recognition of polyhedrons with a range finder. In *Proceedings of the 2nd International Joint Conference on Artificial Intelligence*, pages 80–87, 1971.

- [30] Yoshiaki Shirai. Recognition of polyhedrons with a range finder. *Pattern Recognition*, 4:243–250, 1972.
- [31] W.E.L. Grimson. *Object Recognition By Computer*. MIT Press, 1990.
- [32] O.D. Faugeras and M. Hebert. A 3-d recognition and positioning algorithm using geometrical matching between primitive surfaces. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, pages 996–1002, 1983.
- [33] O.D. Faugeras and M. Hebert. The representation, recognition, and locating of 3-d objects. *Intl. Jour. Rob. Res.*, 5(3):27–52, Fall 1986.
- [34] M. Oshima and Y. Shirai. Object recognition using three-dimensional information. *IEEE Trans. PAMI*, PAMI-5(4):353–361, July 1983.
- [35] T. Fan, G. Medioni, and R. Nevatia. Recognizing 3-d objects using surface descriptions. *IEEE Trans. PAMI*, 11(11):1140–1157, November 1989.
- [36] W.Y. Kim and A.C. Kak. 3-d object recognition using bipartite matching embedded in discrete relaxation. *IEEE Trans. PAMI*, 13(3):225–251, March 1991.
- [37] C.H. Chen and A.C. Kak. A robot vision system for recognizing 3-d objects in low-order polynomial time. *IEEE Trans. SMC*, 19(6):1535–1563, Nov/Dec 1989.
- [38] C. Hansen and T.C. Henderson. Cagd-based computer vision. *IEEE Trans. PAMI*, 11(11):1181–1193, November 1989.
- [39] Peter C. Gaston and T. Lozano-Perez. Tactile recognition and localization using object models: The case of polyhedra on a plane. *IEEE Trans. PAMI*, 6(3):257–266, May 1984.
- [40] W. Eric L. Grimson. The combinatorics of local constraints in model-based recognition and localization from sparse data. *ACM Comp. Surv.*, 33(4):658–686, October 1986.
- [41] P.J. Flynn and A.K. Jain. Bonsai: 3-d object recognition using constrained search. *IEEE Trans. PAMI*, 13(10):1066–1075, October 1991.

- [42] R.C. Bolles and P. Horaud. 3dpo: A three-dimensional part orientation system. *Intl. Jour. Rob. Res.*, 5(3):3–26, Fall 1986.
- [43] Robert C. Bolles and Ronald A. Cain. Recognizing and locating partially visible objects: The local-feature-focus method. *Intl. Jour. Rob. Res.*, 1(3):57–82, Fall 1982.
- [44] Katsushi Ikeichi. Precompiling a geometrical model into an interpretation tree for object recognition in bin-picking tasks. In *DARPA Image Understanding Workshop*, pages 321–339, 1987.
- [45] Katsushi Ikeichi. Generating an interpretation tree from a cad model for 3d-object recognition in bin-picking tasks. *Intl. Jour. Comp. Vis.*, 1:145–165, 1987.
- [46] Daniel P. Huttenlocher and Shimon Ullman. Object recognition using alignment. In *Proceedings of the First International Conference on Computer Vision*, pages 102–111, 1987.
- [47] Daniel P. Huttenlocher and Shimon Ullman. Recognizing solid objects by alignment with an image. *Intl. Jour. Comp. Vis.*, 5(2):195–212, 1990.
- [48] M.A. Fischler and R.C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Comm. of the ACM*, 24(6):381–395, June 1981.
- [49] J. Illingworth and J. Kittler. A survey of the hough transform. *CGVIP*, 44:87–116, 1988.
- [50] W. Eric L. Grimson and Daniel P. Huttenlocher. On the sensitivity of the hough transform for object recognition. *IEEE Trans. PAMI*, 12(3):255–274, 1990.
- [51] P.V.C. Hough. Method and means for recognizing complex patterns. U.S. Patent 3,069,654, Dec. 18 1962.
- [52] Richard O. Duda and Peter E. Hart. Use of the hough transform to detect lines and curves in pictures. *Graphics and Image Processing*, 15(1):11–15, 1972.

- [53] D.H. Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pat. Rec.*, 13(2):111–122, 1981.
- [54] G. Stockman and A.K. Agrawala. Equivalence of hough curve detection to template matching. *Comm. of the ACM*, 20(11):820–822, November 1977.
- [55] Efi Cohen and Haim J. Wolfson. Partial matching of 3-d objects in cad/cam systems. In *Proceedings of the 12th IAPR International Conference on Pattern Recognition*, pages 483–487, 1994.
- [56] Gill Barequet and Micha Sharir. Partial surface and volume matching in three dimensions. In *Proceedings of the 12th IAPR International Conference on Pattern Recognition*, pages 610–614, 1994.
- [57] Andrew Edie Johnson and Martial Hebert. Surface registration by matching oriented points. In *IM97 : Proceedings of the International Conference on Recent Advances in 3-D Digital Imaging and Modeling*, pages 121–128, May 1997.
- [58] Andrew Edie Johnson and Martial Hebert. Object recognition by matching oriented points. In *Proceedings of Computer Vision and Pattern Recognition*, pages 684–689, May 1997.
- [59] Andrew Edie Johnson and Martial Hebert. Efficient multiple model recognition in cluttered 3-d scenes. In *Proceedings of Computer Vision and Pattern Recognition*, pages 671–676, May 1998.
- [60] Chin Seng Chua and Ray Jarvis. Point signatures: A new representation for 3d object recognition. *Intl. Jour. Comp. Vis.*, 25(1):63–85, 1997.
- [61] Roger N. Nagel and Azriel Rosenfeld. Ordered search techniques in template matching. *Proceedings of the IEEE*, 60:242–244, February 1972.
- [62] Avraham Margalit and Azriel Rosenfeld. Using probabilistic domain knowledge to reduce the expected computational cost of template matching. *CGVIP*, 51:219–233, 1990.

- [63] Daniel I. Barnea and Harvey F. Silverman. A class of algorithms for fast digital registration. *IEEE Transactions on Computers*, C-21(2):179–186, February 1972.
- [64] C. Munteanu. Evaluation of the sequential similarity detection algorithm applied to binary images. *Pat. Rec.*, 13(2):167–175, 1981.
- [65] Gordon J. Vanderbrug and Azriel Rosenfeld. Two-stage template matching. *IEEE Transactions on Computers*, C-26(4):384–393, April 1977.
- [66] Azriel Rosenfeld and Gordon J. Vanderbrug. Coarse-fine template matching. *IEEE Trans. SMC*, SMC-7:104–107, February 1977.
- [67] Avraham Margalit and Azriel Rosenfeld. Using feature probabilities to reduce the expected computational cost of template matching. *CGVIP*, 52:110–123, 1990.
- [68] Robert Y. Wong and Ernest L. Hall. Sequential hierarchical scene matching. *IEEE Transactions on Computers*, C-27(4):359–366, April 1978.
- [69] Steven L. Tanimoto. Template matching in pyramids. *Computer Graphics and Image Processing*, 16:356–369, 1981.
- [70] P. Lipson, A.L. Yuille, D. O’Keeffe, J. Cavanaugh, J. Taaffe, and D. Rosenthal. Deformable templates for feature extraction from medical images. *ECCV90 : European Conference on Computer Vision*, pages 413–417, 1990.
- [71] A. Hill, C.J. Taylor, and T. Cootes. Object recognition by flexible template matching using genetic algorithms. *ECCV92 : European Conference on Computer Vision*, pages 852–856, 1992.
- [72] Margrit Betke and Nicholas C. Makris. Fast object recognition in noisy images using simulated annealing. *Proceedings of the Fifth International Conference on Computer Vision*, pages 523–530, 1995.
- [73] H. K. Ramapriyan. A multilevel approach to sequential detection of pictorial features. *IEEE Transactions on Computers*, 1:66–78, January 1976.

- [74] Prabir Bhattacharya and Kai Qian. An algebraic approach for developing vision systems. *Proceedings of the 1991 International Conference on Systems, Man, and Cybernetics*, 1:35–37, October 1991.
- [75] Timothy S. Newman, Anil K. Jain, and H.R. Keshavan. 3d cad-based inspection i : Coarse verification. *Proceedings of the 11th International Conference on Pattern Recognition*, 1:49–52, August 1992.
- [76] Timothy S. Newman and Anil K. Jain. Bidirectional template-matching for 3d cad-based inspection. *Proceedings of the SPIE*, 2183:257–265, 1994.
- [77] Kenneth M. Dawson and David Vernon. Model-based 3-d object recognition using scalar transform descriptors. *Proceedings of the SPIE*, 1609:262–273, 1991.
- [78] J.W. Snell, M.B. Merickel, J.M. Ortega, J.C. Goble, J.R. Brookeman, and N.F. Kassell. Model-based boundary estimation of complex objects using hierarchical active surface templates. *Pat. Rec.*, 28(10):1599–1609, 1995.
- [79] Paul J. Besl and Neil D. McKay. A method for registration of 3d shapes. *IEEE Trans. PAMI*, 14(2):239–256, February 1992.
- [80] Charles W. Therrien. *Decision Estimation and Classification*. John Wiley & Sons, Inc., 1989.
- [81] Tzay Y. Young and Thomas W. Calvert. *Classification, Estimation and Pattern Recognition*. American Publishing Company Inc., New York, 1974.
- [82] Bernard M.E. Moret. Decision trees and diagrams. *ACM Comp. Surv.*, 14(4):593–623, December 1982.
- [83] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees*. Wadsworth Inc., Belmont California, 1984.
- [84] B.M.E. Moret, M.G. Thomason, and R.C. Gonzalez. The activity of a variable and its relation to decision trees. *ACM Transactions on Programming Languages and Systems*, 2(4):580–595, October 1980.

- [85] Donald E. Knuth. *The Art of Computer Programming*, volume 1. Addison-Wesley Publishing Company, 1973.
- [86] Mark W. Kurzyński. The optimal strategy of a tree classifier. *Pat. Rec.*, 16(1):81–87, 1983.
- [87] W. Eric L. Grimson. The combinatorics of object recognition in cluttered environments using constrained search. *Artificial Intelligence*, 44:121–165, 1990.
- [88] Allen Gersho. Principles of quantization. *IEEE Trans. Circuits and Systems*, CAS-25(7):427–436, July 1978.
- [89] Dana H. Ballard and Christopher M. Brown. *Computer Vision*. Prentice-Hall Inc., 1982.
- [90] H. Samet. The quadtree and related hierarchical data structures. *ACM Comp. Surv.*, 16(2):187–260, 1984.
- [91] W.E. Lorensen and H.E. Cline. Marching cubes : A high resolution 3d surface reconstruction algorithm. In *Computer Graphics : Siggraph '87 Conference Proceedings*, volume 21, pages 163–169, July 1987.
- [92] Gerhard Roth and Eko Wibowoo. An efficient volumetric method for building closed triangular meshes from 3-d image and point data. In *Graphics Interface '97*, pages 173–180, May 1997.
- [93] Régis Houde, Jacques Tremblay, Frédéric Auclair, and Denis Laurendeau. Pose determination using octrees : A comparative survey. In *Vision Interface 95*, pages 220–227, 1995.
- [94] Gill Barequet and Micha Sharir. Partial surface and volume matching in three dimensions. *IEEE Trans. PAMI*, 19(9):929–948, 1997.
- [95] Computers & graphics : Special issue on 3d voxel-based graphics, 1989.
- [96] S.M. Goldwasser and R.A. Reynolds. Real-time display and manipulation of 3-d medical objects: The voxel processor architecture. *CGVIP*, 39:1–27, 1987.

- [97] Marc Rioux. Laser range finder based on synchronized scanners. In Thierry Bosch and Marc Lescure, editors, *Analysis and Interpretation of Range Images*. Springer-Verlag, 1990. Also, *Appl. Opt.* 23(21): 3837-3844; 1984, NRC 24528.
- [98] K.W. Bowyer and C.R. Dyer. Aspect graphs : An introduction and survey of recent results. *SPIE Close-Range Photogrammetry Meets Machine Vision*, 1395:200–208, 1990.
- [99] Philipe Brou. Using the gaussian image to find the orientation of objects. *Intl. Jour. Rob. Res.*, 3(4):89–125, Winter 1984.
- [100] William Rucklidge. Efficient guaranteed search for gray-level patterns. In *Proceedings of the 1997 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 717–723, 1997.
- [101] Bernard M.E. Moret. *The Representation of Discrete Functions by Decision Trees: Aspects of Complexity and Problems of Testing*. PhD thesis, University of Tennessee, 1980.
- [102] M.R. Garey and R.L. Graham. Performance bounds on the splitting algorithm for binary testing. *Acta Inf.*, 3:347–355, 1974.
- [103] Richard Bellman. *Dynamic Programming*. Princeton University Press, 1957.
- [104] Lyman Ott. *An Introduction to Statistical Methods and Data Analysis*. PWS Publishers, 1984.