BAYESIAN NETWORKS AND UTILITY THEORY FOR THE MANAGEMENT OF UNCERTAINTY AND CONTROL OF ALGORITHMS IN VISION SYSTEMS

A Dissertation Presented

by

MAURíCIO MARENGONI

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

February 2002

Computer Science

BAYESIAN NETWORKS AND UTILITY THEORY FOR THE MANAGEMENT OF UNCERTAINTY AND CONTROL OF ALGORITHMS IN VISION SYSTEMS

A Dissertation Presented

by

MAURíCIO MARENGONI

Approved as to style and content by:

Allen Hanson, Chair

Edward Riseman, Member

Shlomo Zilberstein, Member

Joseph Horowitz, Member

W. Bruce Croft, Department Chair
Computer Science

To Bruna and Rafael who will, probably, never read it

ACKNOWLEDGMENTS

This is probably the hardest part to write in a thesis. If something is missed in the inside almost nobody will realize it, but if I missed somebody in this part, this will be certainly noticed and hard to repair.

I want, first, to thank my friend and advisor Professor Al Hanson, who helped me throughout this work. We have spent a lot of time discussing the topics presented here and finding a good strategy to run experiments and explaining results, talking about the UMASS basketball team, and playing some basketball here and there. Then I want to thank Professor Ed Riseman for his patience helping me to write a better document and giving me a lot of hints on this final version (sorry Ed that latex does not allow the changes in the figures and tables positions). Professor Shlomo Zilberstein helped me to understand Utility Theory, and I want to thank him for the precious discussions we had to get the theory implemented accordingly. Professor Joseph Horowitz helped me on formatting the math in the final document and I want to thank him for reading and pointing out the problems to me. I want to thank Professor Chris Brown from University of Rochester and Professor Bruce Draper from Colorado State University at Fort Collins who believed in my potential, who helped me to come to the US for my doctoral degree, and who convinced me to go to UMASS where I could learn how to develop research and put things together.

Of course that developing a research work in a lab like the Vision Lab involves other students and staff members, who helped me with valuable discussions, setting up a nice research environment, fixing bugs, implementing some code, or just chatting about how hard is the life of a grad student. So, I want to thank Chris Jaynes, Frank Stolle, Heddi Plumb, Robert Heller, and Laurie Downey. Also, at the Department

of Computer Science I always had the support from CSCF, especially Terry Kellog, and everybody from the main office; in particular I fondly remember the talks I had with my friend Sharon Mallory. I also want to thank the other professors from the department who taught me classes and helped to build a strong background in many different areas of Computer Science. In particular I want to mention Professors Ramesh Sitaraman and Robbie Moll.

Although life as a grad student, sometimes means no life, that is, we have to work most of the time, there is life beyhond that, and I have to admit that I had some social life while working on my research. The friends and relatives who participated of this life during this period probably don't know how important they were on helping to accomplish this work.

I want to thank first Ricardo and Marcia who believed in my capabilities from the beginning, even without really knowing me. Wagner who studied with me in Rochester and helped me to move to UMASS later on. Patricia who was always there to help with my kids and was always smiling no matter how hard it was. A lot of other friends in Rochester helped me in my first years in US, Peter Von Kenel, Cho Man, Martin, Olac, Barbara, Yatsek, and Thereza.

My friends in Amherst, Pilar and her kids Afranio, Bob and Daniel, Eugene, Sandra and their son Carl, Manfred, Sergio, Jefferson and Elizeth. Without these friends I certainly would not have made it to the end. Back in Brazil my friend Edna helped me to put things together and finish (after a long time) this document.

Finally, but certainly not less important, I want to thank my brother Amauri and his wife Dominique for their support and friendship. Also thanks to Adriana, Dicea and Hilton for their patience with me, and sometimes with my grumpy mood. Of course, special thanks go to my parents for their support, patience and for always believing that I would make it. And, closing this chapter I want to thank my kids Bruna and Rafael and I want to thank Hilcea. There is no words or time to write

everything that I should to thank all of you. I know we had hard times but without your close support, friendship, and love this could not be done.

I want to thank God for giving me health, for making me persistent, and for giving me the ability to learn new things, which allowed me to start and finish this episode of my life.

ABSTRACT

BAYESIAN NETWORKS AND UTILITY THEORY FOR THE MANAGEMENT OF UNCERTAINTY AND CONTROL OF ALGORITHMS IN VISION SYSTEMS

FEBRUARY 2002

MAURíCIO MARENGONI

B.S., FACULDADE DE ENGENHARIA INDUSTRIAL - SBC

M.S., UNIVERSITY OF ROCHESTER

M.S., INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS - SJC

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Allen Hanson

An Image Understanding (IU) system should be able to identify objects in 2D images and to build 3D relationships between objects in the scene and the viewer. The system presented here has a control structure for general purpose image understanding that addresses both the high level of uncertainty in local hypotheses and the computational complexity of image interpretation. The control of vision algorithms is performed by an independent subsystem that uses a set of Bayesian networks and utility theory to compute the expected value of information provided by alternative operators and selects the ones with the highest utility value. Each operator has a cost, which is related to the algorithm complexity associated with the operator. The cost of each operator is considered during the operator's selection process. This control structure was implemented and tested on several aerial image datasets. The results

show that the knowledge base used by the system can be acquired using standard learning techniques and that the value-driven approach to the selection of vision algorithms leads to performance gains. Moreover, the modular system architecture simplifies the addition of both control knowledge and new vision algorithms.

TABLE OF CONTENTS

Page

ACKNOWLEDGMENTS ..... **v**

ABSTRACT ..... **viii**

LIST OF TABLES ..... **xiii**

LIST OF FIGURES ..... **xix**

CHAPTER

1. INTRODUCTION **1**

1.1 Statement of the Problem 3
1.2 Research Goals 4

2. RELATED WORK **7**

2.1 Knowledge Based Vision Systems 7

2.1.1 Vision Systems Using Bayesian Networks 9

2.2 The ASCENDER II System 11

2.2.1 Overview 11
2.2.2 Bayesian Networks Background 15
2.2.3 Utility Theory Background 16

3. REASONING OVER FEATURES IN AN AERIAL IMAGE INTERPRETATION SYSTEM **21**

3.1 Using Bayesian Networks to model objects 21
3.2 Structure, Feature Set and Probability Tables 25
3.3 Datasets 28

4. CONTROLLING AND MAKING DECISIONS IN AN AERIAL IMAGE INTERPRETATION SYSTEM - UNCERTAINTY DISTANCE 34

4.1 Feature Selection 34
4.2 The Recognition Process 38
4.3 System Description 43

4.3.1 How the Ascender II System Works - Snapshots 44

4.4 Experiments and Results 46

4.4.1 Evaluation 57

5. CONTROLLING AND MAKING DECISIONS IN AN AERIAL IMAGE INTERPRETATION SYSTEM - UTILITY THEORY 63

5.1 Reasoning in the Ascender II using Utility Theory 63

5.1.1 Selection of a Visual Operator 64
5.1.2 The Recognition Process 68
5.1.3 New Networks 68
5.1.4 How the Ascender II System Works Using Utility Theory - Snapshots 69

5.2 Experiments Using Utility Theory 77

5.2.1 Results 77

5.3 Adding Cost to the Visual Operators 79

5.3.1 Defining a cost associated with a complexity class 83
5.3.2 Tuning the cost function 85
5.3.3 Running the Ascender II system using cost 89

5.4 Changing Preferences in the Reasoning Subsystem 91

5.4.1 Setting preferences - background 92
5.4.2 Setting preferences - general case 95
5.4.3 Experiments changing preferences 97

5.4.3.1 Making strong preferences 97

6. LEARNING MODELS FOR THE CONTROL STRUCTURE 100

6.1 Learning the structures of the knowledge base 100
6.2 Getting data to learn the Bayesian networks 102
6.3 Experiments and Results 104

6.3.1 Using the learned networks on two new datasets 107

7. CONCLUSIONS 114

7.1 Conclusions 114
7.2 Future Work 116

7.2.1 Related to Vision 117
7.2.2 Related to Bayesian Networks 118
7.2.3 Related to Anytime Systems 118

APPENDICES

A. INTRODUCTION TO BAYESIAN NETWORKS 121

A.1 The Bayesian Network Structure 121
A.2 Forecast example using Bayesian networks 123

A.2.1 Building a Model 123
A.2.2 Answering questions without evidence 125
A.2.3 Handling evidence and answering other questions 126

B. A DECISION MAKING EXAMPLE USING UTILITY THEORY 132

B.1 The concept of utility and utility theory 132
B.2 Building a model for Utility Theory 133

B.2.1 Buy a used car example 133
B.2.2 Dealing with uncertainty 137

BIBLIOGRAPHY 141

LIST OF TABLES

Table **Page**

2.1 The decision table shows all utilities u_{ij} for a decision problem with N mutually exclusive events and M decisions. 17

2.2 The table shows all consequences for N events and M decisions. 18

2.3 The table shows all probabilities p_{ij} such that it is indifferent for the user to take C_{ij} with certainty, or the best consequence available with probability p_{ij}. 19

2.4 The table shows the utility values for all possible consequences in a decision problem with N events and M decisions. 20

4.1 Probability distribution of beliefs in the root node for the level 0 network among the states Building, Parking Lot, Open Field, Single Vehicle and Other for all three data sets. 46

4.2 Average number of calls to knowledge sources in the level 0 network, for different data sets, for all classes (Total column), and by specific classes (remaining columns). 47

4.3 Summary of the recognition process for different data sets. 48

4.4 Mean, maximum and minimum errors, in meters, for the 3D reconstruction in Fort Benning. 56

4.5 Number of regions correctly identified and total number of regions at each level for the Fort Benning data set. 58

4.6 Average number of operators called for each region for the Fort Benning data set. 58

4.7 Average processing time (in seconds) on each region for Ascender II and All Evidence methods for the Fort Benning data. 58

4.8 The table presents the results of the 5 random runs executed in the Fort Benning data set. 61

5.1 The table shows all utilities for the level 0 network in the Ascender II system. 65

5.2 The decision table shows all utilities for the level 0 network in the Ascender II system and the initial beliefs for each class in the root node. 72

5.3 The utility value of each decision before acquiring any evidence for the parking lot of boats in the Avenches data set. 72

5.4 Planar Fit. 72

5.5 The expected utility value of each feature. 72

5.6 The beliefs for each class in the root node after evidence for average height equals to 1.1 meters is entered and propagated through the network. 73

5.7 The expected utility value of each feature after the measure of feature Height. 75

5.8 The beliefs for each class in the root node after evidence for height and planar fit (good planar fit = 0.41) are entered and propagated through the network. 75

5.9 The expected utility value of each remaining feature after the measurements of Height and Planar Fit. 75

5.10 The decision table shows all utilities in the level 1 network for Parking Lots in the Ascender II system, and the initial beliefs (a-priori probabilities) for each class in the root node. 75

5.11 The expected utility value of each feature. 76

5.12 The beliefs for each class in the root node after evidence for the feature "Area" with value 1133 square meters is entered and propagated through the network. 76

5.13 The expected utility value of each feature. 77

5.14 Probability distribution of beliefs in the root node for the level 0 network among the states Building, Parking Lot, Open Field, and Other for all three data sets. 78

5.15 Probability distribution of beliefs in the root node of the level 1 network for Parking Lots among the states Parking Lot, Truck-RV, Single Vehicle, and Other for all three data sets. 78

5.16 Probability distribution of beliefs in the root node of the level 2 network for Rooftops among the states Flat, Peak, Cylinder, Flat-peak and Other for all three data sets. 78

5.17 Average number of calls to vision operators. 79

5.18 Summary of the recognition process for different data sets. 79

5.19 Total number of calls to visual operators for all data sets for all classes. 79

5.20 List of visual operators, their corresponding complexity classes, and a short description of each of them. 82

5.21 This table shows the generic function of an operator's cost based on the operator's complexity class. 85

5.22 This table shows the function of an operator's cost based on the operator's complexity class. 88

5.23 Cost for each complexity class and for each average gain used. 91

5.24 Number of operators used in each data set when the average gain changes. 91

5.25 Number of regions correctly classified in each data set when the average gain changes. 91

5.26 The modified utility table shows new values for utilities in the level 0 network. 97

5.27 Comparison in terms of number of regions correctly classified per object class and number of operators used per object class (in parentheses). 97

5.28 The table shows the new utilities for the level 0 network, with a preference for Parking Lot areas. Moderate 2-fold preference. 98

5.29 The table shows the new utilities for the level 0 network, with a strong preference for Parking Lot areas. Strong 10-fold preference. 98

5.30 Comparison in terms of number of regions correctly classified per object class and number of operators used per object class (in parentheses). 98

6.1 Total number of calls to visual operators for the three basic datasets for all classes. 106

6.2 Summary of the recognition process for different data sets using the learned networks. 106

6.3 Probability distribution of beliefs in the root node for the level 0 network among the states Building, Parking Lot, Open Field, and Other for the Flat Scene and Glandorf data sets. 108

6.4 Probability distribution of beliefs in the root node for the level 1 network for Buildings between the states Multilevel and Single Level for the Flat Scene and Glandorf data sets. 108

6.5 Probability distribution of beliefs in the root node for the level 1 network for Parking Lots among the states Parking Lot, Truck-RV, and Single Vehicle for the Flat Scene and Glandorf data sets. 109

6.6 Probability distribution of beliefs in the root node for the level 2 network for Buildings between the states Flat Roof and Peak Roof for the Flat Scene and Glandorf data sets. 109

6.7 Summary of the recognition process for the Flat Scene dataset using the hand-crafted and the learned networks with utility theory and adjusted priors. 109

6.8 Summary of the recognition process for the Glandorf Scene dataset using the hand-crafted and the learned networks with utility theory and adjusted priors. 109

A.1 The prior probability table for rain. 124

A.2 The prior probability table for the sprinkler on. 124

A.3 The conditional probability table of the grass in Dr Watson's house being wet given the knowledge of rain. 125

A.4 The conditional probability table of the grass in Mr Holmes' house being wet given the knowledge of rain and the sprinkler on. 125

A.5 The joint probability table for Mr Holmes lawn, Rain occurred and Sprinkler on. 126

A.6 Probability of the grass in Mr Holmes' house being wet. 127

A.7 The joint probability table for Rain occurred and Dr Watson lawn being wet. 127

A.8 Probability of Dr Watson lawn being wet and dry. 127

A.9 Table for the joint variables Rain occurred, Sprinkler on and Mr Holmes lawn when evidence about the grass being wet is entered. 128

A.10 Normalized table for the joint variables Rain occurred, Sprinkler on and Mr Holmes lawn when evidence about the grass being wet is entered. 128

A.11 Posterior probabilities for Rain occurred given that Mr. Holmes lawn is wet. 128

A.12 Posterior probability for the Sprinkler on given that Mr. Holmes lawn is wet. 129

A.13 Posterior probability table for the variables Rain occurred and Dr Watson lawn being wet. 129

A.14 Posterior probability table for Dr Watson lawn. 129

A.15 Probability table showing evidence that Dr Watson lawn is not wet. 130

A.16 The new posterior joint probability for Rain occurred, Sprinkler on and Mr Holmes lawn. 130

A.17 Probability table when evidence is entered saying that Dr Watson lawn is not wet. 130

A.18 Posterior joint probability table for the variables Rain occurred, Sprinkler on and Mr Holmes lawn being wet given evidence that Dr Watson lawn is wet. 131

B.1 Consequence table for the "buy a used car" problem. 134

B.2 Probability and , in this case utility table for the "buy a used car" problem. 136

B.3 Decision table for the "buy a used car" problem. 136

B.4 Expected utility for each decision in the "buy a used car" problem. 137

B.5 Utility table for the buy a used car problem with a test selection. 138

B.6 Conditional probability table for the Test 1 case. 138

B.7 Conditional probability table for the Test 2 case. 138

B.8 Prior probabilities for the car's current condition. 139

B.9 Summary of the computation of the expected utility of Test 1. 140

LIST OF FIGURES

Figure **Page**

1.1 Different types of regions extracted from aerial images. 2

2.1 Process overview. Decisions are based on current knowledge about the site. 13

2.2 Different types of buildings extracted from aerial images. 14

3.1 The controller starts at level 0 and determines an outcome for the root node at that level (in this case A, B or C). 24

3.2 Bayesian network general structure in the Ascender II reasoning subsystem. 26

3.3 The new general structure for the Bayes Networks in the reasoning subsystem for Ascender II. 29

3.4 Area used for experiments in the Fort Hood data set. 30

3.5 Image 5883 from the Avenches data set. 30

3.6 The MOUT site in the Fort Benning data set. 31

3.7 The left view of the Flat Scene data set. 32

3.8 The Red band of the left view of the Glandorf data set. 32

4.1 The graph shows the uncertainty distance measure for a node with 5 states. 36

4.2 The graph shows entropy (solid line) and uncertainty distance (dotted line) for the state Yes in a Boolean variable (see discussion in text). 37

4.3 Number of operators vs. threshold k. 40

4.4 Regions correctly classified vs. threshold k 41

4.5 Number of times the system did not decide based on the threshold k criterion. 42

4.6 The level 0 network determines if a region belongs to one of the possible object classes: Building, Parking Lot, Open Field, Single Vehicle, and Other. 44

4.7 The level 1 network is invoked for each building found in level 0 to determine if it is a single-level building or a multilevel building. 45

4.8 This network is invoked at level 2. It is called after a single-level building is detected and it is used to determine the rooftop type. 46

4.9 This network is also invoked at level 2. It is called when a multilevel building is detected. The reasoning system calls this network to confirm the recognition at level 1. 47

4.10 Regions obtained by the Ascender I system from the Fort Benning dataset. 48

4.11 Regions obtained from SAR data from the Vexel Corp. 49

4.12 Regions to be identified by the Ascender II system from Fort Benning dataset. 50

4.13 Decomposition of the building hypothesis. 51

4.14 Decomposition of the building hypothesis. 52

4.15 Decomposition of the building hypothesis. 53

4.16 The input regions from the Fort Hood data set. 54

4.17 The input regions from the Avenches data set. 55

4.18 Recognition results on the Fort Hood data. 56

4.19 Recognition results on the Avenches data set. 57

4.20 Recognition results on the Fort Benning data set. 59

4.21 The shadow on the flat roof of this building makes a strong evidence for a center line and the roof top was misclassified as a peaked roof building. 60

4.22 3D reconstruction on the Fort Hood data. 62

4.23 3D reconstruction on the Fort Benning data. 62

5.1 A generic Bayesian network for the Ascender II system. 67

5.2 The level 0 network determines if a region belongs to one of the possible classes. 69

5.3 The level 1 network is invoked for each building found in level 0 and tries to determine if it is a single-level building or a multilevel building. 70

5.4 The level 1 network is invoked for each parking lot found in level 0 and tries to refine its classification. 71

5.5 This network is called after a single building is detected and it is used to determine the rooftop type. 73

5.6 This network is called when a multilevel building is detected. The reasoning system calls this network to confirm the recognition at level 1. 74

5.7 The boat parking lot in the Avenches dataset is used to illustrate how utility theory can be used for feature selection and recognition. 76

5.8 The graph shows the time required to compute line density in the Avenches data set. 81

5.9 The graph shows how the gain in the utility of the system decreases as a function of when an operator is applied. 84

5.10 The graph shows the number of operators used by the system in the Avenches data set when the value of F goes from 0 to 1. 86

5.11 The graph shows the number of operators used by the system in the Avenches data set when the constant B in the cost function for the O(L) operators goes from 0.02 to 1.5. The overall classification did not change. 87

5.12 The graph shows the number of operators used by the system in the Avenches data set when the average gain goes from 1.1 to 10. 88

5.13 The graph shows the average beliefs for each level in the Avenches data set when the average gain goes from 1.1 to 10. 89

5.14 The graph shows the number of regions correctly classified in the Avenches data set when the average gain goes from 1.1 to 10. 90

6.1 A building region in the Avenches data set. 103

6.2 The level 0 network learned from data determines if a region belongs to one of the possible object classes: Building, Parking Lot, Open Field, or Other. 104

6.3 The level 1 network for buildings learned from data determines if a region classified as a building is a single level or a multilevel building. 105

6.4 The level 1 network for parking lots learned from data determines if a region classified as a parking lot is a real parking lot, an RV-truck, or a single vehicle. 107

6.5 This level 2 learned network is called after a single building is detected. 108

6.6 Set of regions extracted by hand from the Flat Scene dataset. 110

6.7 Set of regions extracted by hand from the Glandorf dataset. 111

6.8 Classified regions for the Flat Scene dataset. 112

6.9 Classified regions for the Glandorf dataset. 113

7.1 Average weighted utility in the Ascender II system. 120

A.1 Simple Bayesian network with corresponding probability tables. 122

A.2 Simple Bayesian network. 124

CHAPTER 1

INTRODUCTION

Image Understanding (IU) systems are defined as systems capable of identifying objects in 2D images and of building 3D relationships between objects in the scene and the viewer. The construction of an end-to-end, general purpose, image understanding system that is capable of supporting an intelligent agent in different domains is one of the main goals of computer vision. Significant progress has been achieved on domain specific systems and/or isolated modules of a general system, but the main goal is still out of reach. A large number of image understanding systems developed so far are dedicated to Aerial Image Interpretation. One of the problems with aerial image interpretation systems is the management of uncertainty. Uncertainty in this case arises from a variety of sources, such as the type of the sensor (optical, infrared, radar), the sensor's quality, the sensor's position, weather conditions (humidity, cloud conditions), illumination conditions (the angle of the sun), season (spring, summer, fall or winter), random objects in the scene (number of vehicles or persons), and the inherent uncertainty in the definition of common objects. A vision system for aerial image interpretation has to be able to recognize and reconstruct objects in a scene and to deal with many levels of uncertainty.

Object recognition in aerial images is one important step towards 3D reconstruction of a scene, but to automate the recognition process in a real world application is not an easy task. Consider the tiles extracted from larger aerial images presented in Figure 1.1

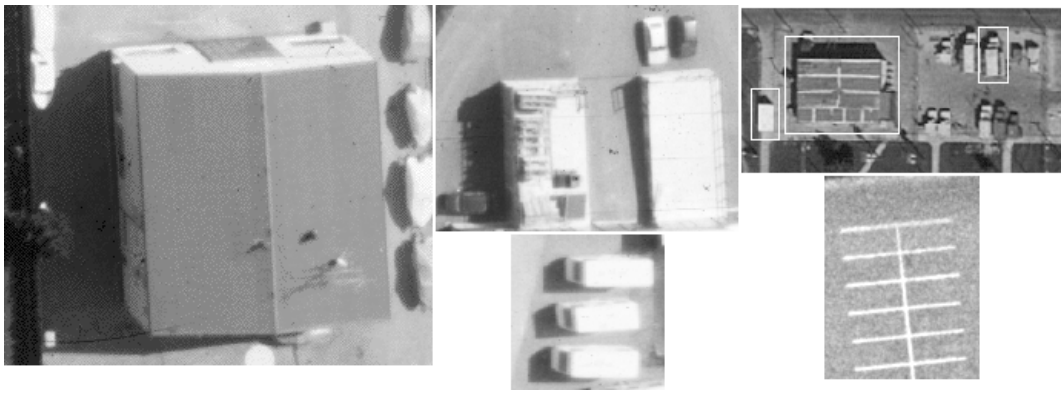

Figure 1.1. Different types of regions extracted from aerial images.

The tile at left contains a building, which is easy to identify by looking at its general shape, the door, and the rooftop. A more difficult task is to identify the two large objects in the top middle tile. Comparing the shadows of these areas with the shadows of the cars present in the image, it is possible to say that the height of the two areas is greater than the height of the cars. These two areas look like buildings, but they are not as easy to identify as the building in the first tile. The bottom middle tile has three objects. It is hard to say what they are at first. If it is known that this tile came from the same image as the previous two, it is possible to infer, from the size of the objects, and other sparse cues (such as the windshields), that they are Recreational Vehicles (RV's) or some other kind of delivery truck. The recognition of the three objects marked in the top right tile are not as simple, and more comparisons and measurements may be required to identify them correctly. The region at right (outlined by a box) is a big truck. The lines on top of the large region to the left of the middle could lead a reasoning process to identify the region as a parking lot (compare it with the bottom right tile, which is a real parking lot). A more careful inspection of this region using an analysis of the shadow and studying other objects close to the region would help to identify this region as a building. The region at left could be identified either as a small building or as a large vehicle, such as a bus. The

region's features (shadow, area, shape and appearance) do not help in discriminating between these two hypotheses. The true identity of this region is unknown.

The research presented here addresses some of the control issues involved in the interpretation process for a general purpose image understanding system that could reason about the types of interpretation problems just discussed. These issues are related to the construction of efficient knowledge bases, the selection of appropriate visual operators given the knowledge available, management of uncertainty arising from both the data and the operators, and fusion of information from different sources which can either support or refute a certain hypothesis.

1.1 Statement of the Problem

The interpretation of an image can be viewed as establishing a correspondence between image features and the identified object classes. It is clear that the descriptive vocabulary of the system must be reflected in the set of features extractable from the image. Thus, the image features must form the primitive descriptions of the objects in the knowledge base. Since every feature has at least one visual operator for measuring it, the control problems we address in this research are these: (i) given a general purpose system and a specific interpretation problem within the domain of the system, how do we effectively select the features to measure or, more generally, decide which visual operator to apply, and in what order, (ii) how do we know when to stop the reasoning process, and (iii) what class label should we give to that region. Furthermore, because there is a significant amount of inherent ambiguity in the interpretation process, an interpretation system must include a sufficiently rich set of relations among features as well as flexible mechanisms for manipulating uncertain hypotheses until there is a convergence of evidence.

In this thesis it is shown how Bayesian networks and utility theory are used to build a control structure for a general purpose image understanding system. It also

addresses the knowledge engineering issue by demonstrating that it is possible to learn the Bayesian network structures from fairly coarse training information. Ascender II, an IU system for fully automated Aerial Image Interpretation, is used as a testbed to address these questions:

- How can the results of visual operators and their associated uncertainties be combined in order to classify a particular image region?
- How can the hierarchical structure of objects be exploited in order to construct an incremental classification process?
- Can the construction of the knowledge base be simplified (or fully automated) for a particular application using both human expertise and machine learning techniques?
- Can performance be improved by using a disciplined approach to operator selection?

1.2 Research Goals

In this thesis we discuss important issues related to reasoning in an image understanding system. The ideas discussed here are implemented in an aerial image interpretation system called Ascender II. The Ascender II system is an ongoing project that was first developed for the detection and reconstruction of buildings from aerial images, but it has been generalized to other object classes such as parking lots and vehicles. Some of the contributions of the work presented in this thesis include the following:

- The knowledge base and control processes (reasoning subsystem) are completely separated from the visual subsystem (visual operators, models, images). The separation allows more flexibility in adding new visual operators, or in replacing current operators with only minor changes in the reasoning subsystem [48].

- A set of Bayesian networks is used to implement the knowledge base underlying the reasoning subsystem for a real world application. Bayesian networks combine information from different sources, and have been used successfully in selective perception systems applied in well-controlled environments [53].

- The knowledge base is divided into levels of detail and reasoning is performed within each level. The hierarchy used in the knowledge base avoids large Bayesian networks, which leads to faster propagation of beliefs in the networks.

- The use of the hierarchical structure also allows local reasoning. The drawback of local reasoning is that, if backtracking is required, it has to be performed using a pre-defined strategy. We believe that if the models are developed coherently, backtracking will not be required (although we leave this as an open research issue).

- Each network can be used to test the efficiency of visual operators. This can be done before the operator is implemented. The visual operator can be simulated in a Bayesian network just by adding a node and a link. The link has a conditional probability table which represents the operator's reliability. The values in the conditional probability table are changed during the simulation up to the point where the operator has an impact in the decision process. The simulation will show how reliable the operator has to be in order to be useful for the system.

- The application in aerial image interpretation suggests that Bayesian networks are robust in the face of either redundant or conflicting information and, if sufficient information and time for processing is available, the system converges to a correct outcome.

- Utility theory is a well-defined method for making decisions when dealing with uncertain events. It is based on probability theory, which fits well in the environment described above. Utility theory can be used to select the most appropriate visual operator from a library of visual operators, and to decide about the correct outcome in the recognition process based on a more complete information about the state of the system. This will be contrasted with an alternative heurisitic approach defined in this work, called uncertainty distance.

A discussion of previous work on image understanding systems and some background on the ASCENDER II system are presented in the next chapter. Chapter 3 presents the reasoning subsystem, its knowledge base, and details about the Bayesian network structures, as well as a description of the datasets of images used in this thesis. Chapter 4 and 5 present issues related to decision making and management of uncertainty using uncertainty distance and utility theory respectively; they also describe the results of experiments performed on the image datasets. Chapter 6 shows that the structure of the networks and/or the conditional probability tables can be learned from data. Results of experiments using networks learned from data are also presented in this chapter. Finally, in chapter 7, conclusions from this research are presented as well as a set of interesting research questions related to extending the current framework.

CHAPTER 2

RELATED WORK

2.1 Knowledge Based Vision Systems

One popular approach to the general Image Understanding problem in the 1980's was knowledge-directed vision systems. The typical knowledge-directed approach to image interpretation seeks to identify objects in unconstrained two-dimensional images and to determine the three-dimensional relationships between these objects and the camera by applying object- and domain-specific knowledge to the interpretation problem. A survey of this line of research in computer vision can be found in [19], [22], and [26].

Typically, a knowledge-based vision system contains a knowledge base, a controller, and visual operators (refered to, in the literature, as knowledge sources, IU processes, algorithms, visual or vision algorithms, vision operators, or simply operators). Knowledge representations range from semantic nets in the VISIONS system [25], and later schemas [21], to frames in the ACRONYM system [7], and rules in the SPAM system [49], to relational structures (generalized models) of objects in the MOSAIC system [30]. Controllers are typically hybrid hierarchical systems, mixing bottom-up and top-down reasoning. The order in which different visual operators are applied is dictated by the control structure. In some heterarchical control systems the order in which different visual operators are applied is dictated by the data, where blackboards are used as a global database. In this case visual operators are triggered if their preconditions have been met and their results were written in the blackboard

for future use. Systems developed using this approach include the ABLS system [59] and the VISIONS schema system [21].

In most of these systems (VISIONS, ACRONYM, SPAM, or MOSAIC) the controller and the visual operators are combined into a single system. However, these systems can not be easily generalized to domains that are different from those for which they were originally developed. Furthermore, the amount of specific knowledge required to use the system in a different domain would be a burden in constructing it.

Some of the problems found with most of the knowledge-directed vision systems are the following: control for visual operators was never properly addressed as an independent problem [22], and the system's structure did not facilitate entry of new knowledge [19]. From the surveys in [19] and [22], an ideal general purpose Image Understanding system should take into account the two issues just described and have, at least, the following characteristics:

- **Multiple levels of representation** to accommodate a diverse set of features and objects. This is provided in our system by hierarchically structured Bayesian networks.

- **Dynamic Planning** to make use of the current context and state of interpretation. Once a specific task for the system has been identified (for example: classify a given region), an ordered sequence of visual operators is generated. This is accomplished in the Ascender II system using measures based on the current state of the knowledge base (e.g., uncertainty distance or utility theory).

- **Control the flow of processing** during analysis. In the Ascender II system the visual operators are invoked based on the outcome of the set of visual operators used. The controller uses updated information to decide what to do next.

- **Map numerical values into symbolic quantities.** This is done by a coherent discretization of the possible outcomes for the features used in the system, and by the relationship between features and object classes represented in the networks.

- **Combine evidence** from different sources. The fusion of information is accomplished in the Bayesian networks.

- **Recognize objects and resolve conflicts** between contradictory hypotheses. This is computed using the Bayesian networks combined with either a threshold technique or utility theory.

2.1.1 Vision Systems Using Bayesian Networks

Bayesian networks have been successfully used in systems required to combine and propagate evidence for and against a particular hypothesis. Vision systems have been developed using Bayesian networks for both knowledge representation and as a basis for information integration.

The TEA1 system [53] was developed using a set of Bayesian networks that are combined to define what visual operator should be used and where the visual operator should be applied to achieve scene interpretations at a minimum "cost". The TEA1 system was designed to interpret table top scenes, and classify them with respect of one of the possible classes available. The controller in the TEA1 system had to deal with operator selection and control of the camera position in an active vision scenario. Although this system used selective perception [8] to reduce the number of visual operators called, the knowledge base encoded domain-specific knowledge and was difficult to construct because of the level of detail required. The knowledge base was composed of five structures: a PART-OF net to model the physical structures of the scene; the Expected Area net which models spatial relationships between objects in the scene; an IS-A tree that models the possible outcomes of a random variable;

the TASK net, used to select operators to answer specific questions; and finally, a Composite network, added to combine all the information from the four networks. Although the classification results were satisfactory, the system was slow, it did not support "real-time" applications, and it was applied in a well-controlled indoor environment [53].

Mann and Binford [46] showed how to make bottom-up inferences using geometric properties and Bayesian aggregation. The system was designed to recognize and reconstruct simple mechanical objects, such as an elbow. This system, called SUCESSOR, uses two networks. The first (object model) represents physical properties of an object, breaking the object into structural parts (e.g., house=walls+roof). The second (aggregation network) represents geometrical relations between parts of an object and geometrical forms that can be extracted from the data. The example presented in the paper was simple and the indoor environment well controlled. Mann's system can be applied only for small sets of objects because a model for each object has to be built manually.

Kumar [43] introduced a system with simple networks (each network has only 2 layers) for aerial image interpretation. In this system, after an initial segmentation step, a Bayesian network is built and a feature vector (area, average grey level, average texture density, and contrast between two areas) is computed from the image. These features are fed into the network and propagated to generate a label for each region. In general the features are simple to compute, but a new network needs to be built for each image. The probability tables used in the network were generated heuristically. Because only one example is shown, it is hard to determine whether the system can be generalized to other images.

More recently Krebs [42] presented a system using 3D B-spline curves as features to identify objects in a well-controlled indoor environment. Kreb's system uses a "belong-to" network that models all objects in the scene and their parts as curves

and sub-curves. The system uses an Influence Diagram ("task net") [32] to control and make decisions. Although the results were promising, the example used was too simple, and it is not clear what would happen if one of the objects present in the scene was not modeled.

2.2 The ASCENDER II System

2.2.1 Overview

The original Ascender system (called Ascender I here) was developed for building detection and reconstruction from multiple aerial images of a site [15]. It used 2D image features and grouping operators to detect rooftop boundaries [37] and then matched these polygons under 3D constraints and across views to compute height and to build 3D volumetric models. The system used a fixed strategy and it detected nearly 90% of the buildings in controlled experiments on imagery from Fort Hood, Texas. However, a considerable number of false positives were generated due to scene clutter and the presence of buildings outside of the class for which the system was designed (single flat-roofed buildings) [16].

It was learned from the Ascender I system that the use of multiple strategies and 3D information fusion can significantly extend the range of complex building types that can be reconstructed from aerial images [33]. In order to address this problem of generality, the Ascender II system has been developed to incorporate AI mechanisms for dynamic control of a large set of visual operators, from simple T junction detectors to complex operators such as the Ascender I system used for polygon grouping and flat roof building detection.

The design approach for Ascender II is based on the observation that while many IU techniques function reasonably well under constrained conditions, no single IU method works well under all conditions. Consequently, work on Ascender II is focusing on the use of multiple alternative reconstruction strategies from which the most

appropriate strategies are selected by the system based on the current context. In particular, Ascender II utilizes a wider set of visual operators that fuse 2D and 3D information and that can make use of EO, SAR, IFSAR, and multi spectral imagery during the reconstruction process if available.

The Ascender II system was designed for aerial image interpretation, particularly for the 3D reconstruction of urban areas. The system is divided into two independent parts (the reasoning subsystem and the visual subsystem), in our case running on different operating systems on different machines. The reasoning subsystem has a knowledge base composed of a set of Bayesian networks. This subsystem is responsible for selecting visual operators and for making decisions throughout the inference process. The visual subsystem has a data base of images, models, visual operators and other image features extracted during the inference process. These two subsystems communicate through sockets using a well-defined communication protocol. This framework is presented in Figure 2.1. One advantage of this design is that changes to the reasoning subsystem, or to the visual subsystem, can be made independently of the other. For example, visual operators can be exchanged or augmented in the visual subsystem without changing the reasoning subsystem.

Although the initial effort has focused primarily on recognizing and reconstructing buildings from aerial images, Ascender II has been designed as a general purpose vision system. The system has a set of focus-of-attention regions as input. These regions can be extracted from aerial images automatically (using a system such as Ascender I [15]), manually, or interactively (using cues from other sources such as maps or other classified images). The system's goal is to select visual operators, recognize objects in the scene, and reconstruct these objects automatically in 3D. This whole process should be performed as fast, and as accurately, as possible to help human analysts making decisions about the area being processed.

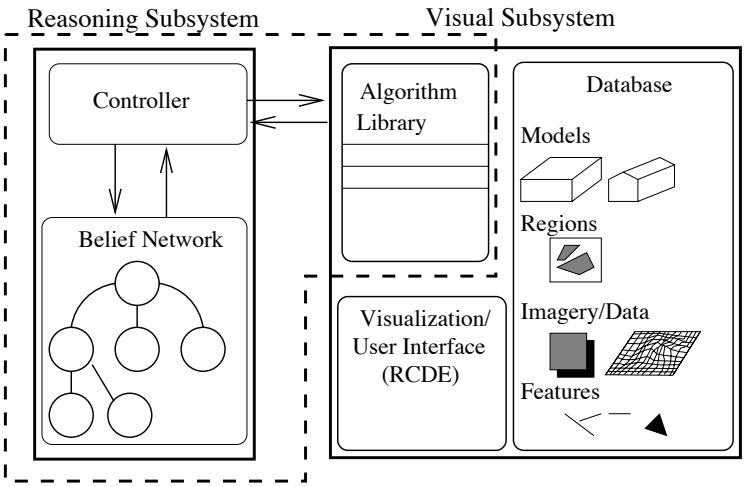

Figure 2.1. Process overview. Decisions are based on current knowledge about the site. Visual operators, stored in the visual subsystem, gather evidence about the site, update the knowledge base, and produce geometric models.

As mentioned earlier, aerial image interpretation is a difficult problem. Even if the focus-of-attention regions are buildings it is impossible to pre-define 3D models for every type of building. Consider the sample of buildings presented in Figure 2.2. The building at left and the building in the middle can fit into a rectangular polygon, but not the building at the right (a rectangular fit would miss part of the building or would add roads and ground area). The building at left is a single level building with a peak type rooftop (see the roof centerline). The building in the middle is also a single level building, but its rooftop is a composite of simple structures. Checking the image it is possible to identify two flat areas (left and right of the white area), and two cylindrical areas (see the shadow of the white area on the left side of the building). The building at right has multiple levels, each with a flat rooftop. If objects in the surrounding area were correctly identified (cars, boats, roads, trees, etc.), more information about the buildings, such as height, area, and usage, could be extracted.

Previous systems for aerial image interpretation, such as ACRONYM [7] and SPAM [49], were typically model-based systems, where all objects expected in the

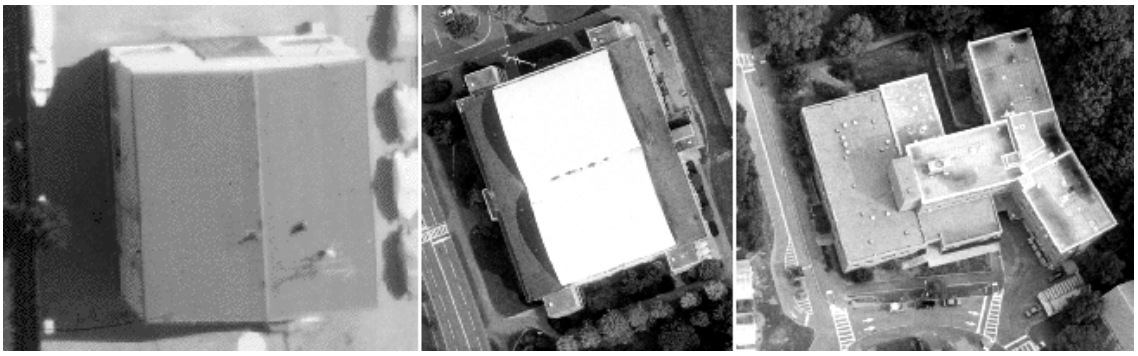

Figure 2.2. Different types of buildings extracted from aerial images.

scene are modeled. These systems used a top-down approach to match regions with the available models. This approach is reasonable only if the number of objects in the knowledge base is small because the models are built manually (as in an application of ACRONYM to identify commercial jet planes in airport scenes [49]). This approach can not be used for buildings because buildings appear in different shapes, levels, colors, with different rooftops, etc.

Another common approach for classification uses primitives combined with a bottom-up inference. In this case the system identifies the primitives in the image and builds up an object model from the primitives found using local constraints, e.g. geometry. A single primitive can be part of a large number of objects, thus generating a large number of hypotheses that have to be confirmed. In aerial image interpretation this approach is possible only if the image resolution is high enough to extract the set of primitive features and the combinatorics of the hypothesize-and-test paradigm can be reasonably constrained.

The Ascender II system combines top-down and bottom-up inference. It uses top-down inference to go from a general interpretation of the region in the image to a detailed interpretation. Bottom-up inference is used within each interpretation level, identifying primitives (features) in the image and using these primitives to classify (recognize) the regions. The use of Bayesian networks and utility theory leads to a system that measures only the most useful subset of the features available. Using

this subset the system is capable of recognizing the regions correctly, saving time and reducing the cost of processing.

The work described here presents the ideas behind the reasoning subsystem (the control process and management of uncertainty) of the Ascender II system. The reasoning subsystem is based on a hierarchy of Bayesian Networks and uses Utility Theory to select visual operators and to make decisions. It will be shown that the Ascender II system has most of the characteristics desired in a general purpose Image Understanding system: multiple levels of representation, dynamic planning, control of the processing flow, mapping of numerical values into symbolic quantities, combining evidence from different sources, resolving conflicts between contradictory hypotheses, and facilitating the entry of new knowledge.

2.2.2 Bayesian Networks Background

As defined by Jensen [38], a Bayesian network is a probabilistic inference method which consists of:

- A set of variables and a set of directed links between variables, where the links represent some type of relationship between the variables.
- Each variable has a finite set of mutually exclusive states.
- The variables and the directed links form a directed acyclic graph (DAG).
- For each variable A with parents $B, C, \cdots, N$, there is an associated conditional probability table that represents $P(A|B, C, \cdots, N)$.

Bayesian network systems have been used as an inference method in many domains such as agriculture [52], computing [28], information processing [31], medicine [3, 29], forecasting [1], and computer vision [39, 57, 10, 55, 51, 9, 6]. The inference process in Bayesian networks can be classified [54] as follows:

Diagnostic inference: moves from effects to causes. That is, knowing the facts available might lead to the most probable cause, e.g. if one verifies that the grass in front the house is wet early in the morning, this might help to conclude that the most probable cause is rain.

Causal inference: goes from causes to effects. For example, knowing that it rained during the night will lead to wet roads and thus traffic jams.

Intercausal inference: goes between causes of common effects. For example, if a traffic jam is caused either by wet roads or traffic accidents and, if there was no rain to make the roads wet, then the belief in a traffic accident will increase.

Mixed inference: a combination of the processes above.

The propagation of beliefs in a Bayesian network, which is the mechanism used to make inferences in the system, is an NP-Hard problem [17]. The algorithms that propagate beliefs and update the beliefs in each node can be exact algorithms (for simple structures) or approximate algorithms (for complex structures); a detailed explanation of the most common algorithms can be found in Castillo [12]. The propagation is also affected by the number of nodes in the network, the density of links in the network, the structure itself and the number of states in each node [54].

Bayesian networks can be used to address the following types of questions [38]: forecast (estimate the state of a variable given evidence), most probable configuration of the variables, checking data conflict, sensitivity analysis, and value of information. Introductions to Bayesian networks can be found in [50, 38, 54]; a simple example of forecast using Bayesian Networks is presented in Appendix A.

2.2.3 Utility Theory Background

Utility theory is the branch of decision theory concerned with measurement and representation of preferences. Researchers in utility theory focus on accounts of pref-

Table 2.1. The decision table shows all utilities u_{ij} for a decision problem with N mutually exclusive events and M decisions.

Decisions	*Events*			
	E_1	E_2	$\cdots$	E_N
Decision 1	u_{11}	u_{12}	$\cdots$	u_{1N}
Decision 2	u_{21}	u_{22}	$\cdots$	u_{2N}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
Decision M	u_{M1}	u_{M2}	$\cdots$	u_{MN}
Probability(Event)	p_1	p_2	$\cdots$	p_N

erences in rational decision making, where preferences cohere with associated beliefs and actions. In this case, utility refers to the scale on which preference is measured. Utility theory has some of its ideas based on social studies performed by Bentham in the 18^{th} century [5], related to pain and pleasure. Today, this technique is applied in different areas such as economics [20, 41, 45], knowledge representation [24, 4], and game theory [58].

Utility theory has a set of principles that lie behind every economic transaction and most non-economic transactions. A typical problem solved using utility theory can be described as follows: consider the decision problem where there are N possible, mutually exclusive events, and a decision, among M possible decisions, has to be made. Utility theory selects the decision that has the highest utility, which is computed using a decision table (see Table 2.1), based on the probability of an event being true and the utility of each decision given that the event is true.

The utility values in the decision table are computed by comparing consequences for each possible outcome (event, decision). Notice that for any practical problem, each pair (event, decision) has a practical consequence related to it. All possible consequences in a problem with N events and M decision are shown in Table 2.2.

Now the consequences have to be ranked from best (most favorable) to worst (least favorable). The ranking can be computed as follows: assume that only one

Table 2.2. The table shows all consequences for N events and M decisions.

Decisions	*Events*			
	E_1	E_2	$\cdots$	E_N
Decision 1	C_{11}	C_{12}	$\cdots$	C_{1N}
Decision 2	C_{21}	C_{22}	$\cdots$	C_{2N}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
Decision M	C_{M1}	C_{M2}	$\cdots$	C_{MN}

of the events is true (consider only one column in Table 2.2), find the best and the worst consequences in the column, and rank all other consequences in the column accordingly; repeat the process for all other events. At this point, it is possible to compare all the best and worst consequences from different columns and find the best and the worst of all consequences in the table. Comparing the other consequences with the best and the worst results in a partial ordering among all consequences in the table, with ties solved randomly.

The next step consists in converting each consequence into a numerical value. The numerical values, or utilities, are computed as follows:

- Define a numerical value, say "C", for the best consequence in the table (any positive value).
- Define a numerical value, say "c", for the worst consequence in the table.
- For all consequences C_{ij} in the table find the corresponding value of p_{ij} as follows:
  - Choose any consequence C_{ij}. C_{ij} is worse, or at most as good as the best consequence; and C_{ij} is better, or at least no worse than the worst consequence.
  - Define the probability p_{ij} using the following methodology: suppose you are given a choice. You can have consequence C_{ij} with certainty, or you can

Table 2.3. The table shows all probabilities p_{ij} such that it is indifferent for the user to take C_{ij} with certainty, or the best consequence available with probability p_{ij}.

Decisions	*Events*			
	E_1	E_2	$\cdots$	E_N
Decision 1	p_{11}	p_{12}	$\cdots$	p_{1N}
Decision 2	p_{21}	p_{22}	$\cdots$	p_{2N}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
Decision M	p_{M1}	p_{M2}	$\cdots$	p_{MN}

gamble to have consequence C with probability p_{ij} and c with probability $1 - p_{ij}$. If $p_{ij} = 1$ you will select to gamble, and if $p_{ij} = 0$ you will select C_{ij}. So find a value of p_{ij} that makes the selection between C_{ij} and the gamble indifferent (see Appendix B for an example). Table 2.3 shows all probabilities for a decision problem with N outcomes and M decisions.

- Compute the numerical utility of C_{ij}, defined as $u(C_{ij})$, using the expression below:

$$u(C_{ij}) = p_{ij} * C + (1 - p_{ij}) * c$$

Table 2.4 shows the utility values for a decision problem with N outcomes and M decisions. Usually events are not known with certainty, but they have an expectation (or prior probabilities). In this case only the expected utility of each decision can be computed. The expected utility of a decision is defined as the sum over all events of the utility of each consequence times the probability of the event:

$$EU(Decision_i) = \sum_{j=1}^{N} u(C_{ij})p(j)$$

The current utility of the decision problem is defined as the maximum value among each of the expected utilities:

Table 2.4. The table shows the utility values for all possible consequences in a decision problem with N events and M decisions.

Decisions	*Events*			
	E_1	E_2	$\cdots$	E_N
Decision 1	$u(C_{11})$	$u(C_{12})$	$\cdots$	$u(C_{1N})$
Decision 2	$u(C_{21})$	$u(C_{22})$	$\cdots$	$u(C_{2N})$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
Decision M	$u(C_{M1})$	$u(C_{M2})$	$\cdots$	$u(C_{MN})$

$$max(EU(Decision_i))$$

The best decision is defined as the decision i which gives the maximum expected utility (the argument of the expression above):

$$argmax_i(EU(Decision_i))$$

A simple example on how to use utility theory in a decision making problem is presented in Appendix B.

CHAPTER 3

REASONING OVER FEATURES IN AN AERIAL IMAGE INTERPRETATION SYSTEM

3.1 Using Bayesian Networks to model objects

The goal of the reasoning subsystem is to accumulate evidence sufficient to determine a plausible identity of a selected image region in terms of the object classes represented within the system. A priori knowledge about objects and their relationships is captured in a hierarchical Bayesian network structure (see below). Each network in the Ascender II system is used to perform the following tasks related to a specific image region: select a feature to be measured in the image, combine the evidence from different features once they have been collected, and decide when to accept the current hypotheses and stop the reasoning process on a region.

Evidence about a feature is obtained by applying a corresponding visual operator to the region in the image. A priori knowledge, in the form of initial prior probabilities associated with each object class, is used to select the image understanding (IU) process for the initial step. Once evidence is obtained, it is entered in the Bayesian network and it is combined with previous knowledge. The process is repeated until the system accumulates enough information to determine the region's most representative object class.

One problem with Bayesian networks is that propagation of evidence is, in general, an NP-hard problem [17]. If the network structure is a tree or a polytree (a structure in which a node can have multiple parents, but there is no closed circuit in the undirected graph underneath), evidence can be propagated in linear time [50].

To avoid the general propagation problem, our reasoning subsystem has been designed using a set of small Bayesian networks organized into a hierarchical structure according to levels of detail. One of the networks is a general graph with an embedded cycle that leads to the NP-Hard propagation problem (see the Level 0 network in Chapter 5). Because the network is small in terms of number of nodes and density of links, the propagation time in this network is not a problem, and does not affect performance. However, we don't know when the NP-Hard propagation problem starts to interfere in terms of processing time, and this analysis is beyond the scope of this thesis.

The hierarchical network structure is the system's knowledge base. Decision procedures use the information inside the networks to decide what to do next. The idea of using this set of networks, designed hierarchically, to control the inference process is new [35], and its drawback is that backtracking in the hierarchy is difficult to support. Backtracking is a term used in systems with the hypothesis-and-test approach and it is the action taken when the test does not confirm a hypothesis and a new hypothesis has to be tested. In a Bayesian network the evidence entered and propagated will, automatically, lead to one or another hypothesis and backtracking is not required. In the hierarchy of Bayesian networks proposed here backtracking would be required when the system classifies a region as one of the object classes available and calls a network in the next level for a detailed classification and the evidence obtained in this network leads to a change in the classification obtained in the previous level. We claim that the system's ability to make local inferences faster while classifying objects in a well-defined hierarchy is a worthwhile tradeoff. If the knowledge base in the Ascender II system was composed of only one network, the backtracking problem would not exist. The experiments presented later show that backtracking was required only twice in 80 regions. In these cases the ambiguity was solved by calling a procedure

hand-crafted in the code. This issue needs further study, which is also beyond the scope of this thesis.

Each network represents knowledge about an object class at a particular scale of detail. Each object class has also been decomposed into subparts within a network, and each subpart may be represented by a separate network. The subparts are basic components of the objects such as rooftops for buildings, vehicles for parking lots, etc. The reasoning process is performed at each level assuming that recognition in the previous stage was correct. As an example, if a building is found in the first level, at the next level the system determines whether the building has a single-level or multiple-level rooftop; if a single-level building is detected, it is possible to recognize the building's rooftop; otherwise it breaks the building into subregions, such that each subregion is a single rooftop, and then classifies their rooftops.

The hierarchy in the knowledge base is designed as follows: a network at level i represents a refinement of a class represented in a network at level $i-1$. This organization of networks is shown in Figure 3.1.

The first level attempts to recognize that a region belongs to a generic class, like building, or grassy field, or parking lot. The second level assumes that the region belongs to the generic class found in the first level and attempts to refine the classification within that same object class; for instance, if a region is recognized as a *parking lot* at level one, it might be recognized as a *full parking lot* at level two, using an IU process that counts vehicles in the region and compares the area covered by cars with the region's area.

The Bayesian networks were developed using the HUGIN system [2]. One of our goals is to show that using small networks, plus the hierarchical structure suggested here, will increase performance and will avoid propagation of evidence through variables that will not affect the overall classification process, because they are not directly related to the immediate local decision.

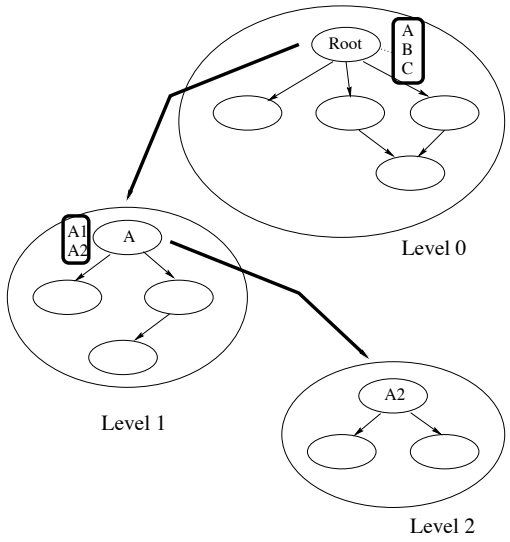

Figure 3.1. The controller starts at level 0 and determines an outcome for the root node at that level (in this case A, B or C). If the outcome is A and time for further computation is available the controller loads the network for A at level 1 (the thicker line shows this inference call). The process is repeated until the last level is reached or the time for computation has expired.

The knowledge in each network is structured as follows: each network has only one root node, and each state of the root node represents a possible class for a region. The other nodes in the network are random variables representing features. The features were selected such that their values are expected to discriminate between two or more classes, or to help confirm that a region belongs to a certain class. The feature set ranges from a simple line that can be found in the optical image to a complex planar fit which requires either a DEM or a stereo matching operator. Each arc represents the relationship between features or between a feature and a class in the root node. Each feature has associated with it one or more visual operators in the visual subsystem. The visual operators are responsible for computing and interpreting their corresponding feature.

Two types of knowledge are encoded in the network: domain-specific knowledge in the form of prior probabilities for each class, and general knowledge which shows the

relationship between a class and a feature, and is represented by conditional probability tables. Changes in the distribution of objects in the data set imply an adjustment to the set of prior probabilities used for each network in the reasoning subsystem. A link from the root node to the feature has associated with it a conditional probability table which represents the conditional probability of the feature F having a value "k" given that the class C has a value "c", described in the expression:

$$P(F = k/C = c)$$

The inference process in Bayesian networks uses Bayes Rule, which allows reasoning in both directions (from cause to consequence and back).

$$P(F = k/C = c) = \frac{P(C = c/F = k) * P(F = k)}{P(C = c)}$$

or

$$P(C = c/F = k) = \frac{P(F = k/C = c) * P(C = c)}{P(F = k)}$$

Thus, a feature can be measured for a region and its value propagated through the network, ultimately changing the beliefs in the classes at the root node for that region.

The hierarchical structure applied in the reasoning subsystem can be used as a basic framework to implement an incremental or "anytime" recognition system. This topic will be left for future work.

3.2 Structure, Feature Set and Probability Tables

Although no constraint was placed on the network structures, they were designed as simply as possible (utilizing a small number of nodes and edges) to avoid a seri-

ous impact from the exponential increase in propagation time. The networks were designed using the following principle:

- The root node represents the region of discourse.
- The states in the root node are the possible classes that the region can belong to.
- The children of the root node are features that can either be measured or inferred from measurements performed on the image.
- Each feature is discretized into ranges such that a feature value can be used to discriminate between the classes in the root node; these ranges were defined empirically, and they are another part of the knowledge engineering process.

The basic structure of the networks is presented in Figure 3.2. Features can be measured directly (like feature 1) or indirectly (measure feature 21 which will give evidence about feature 2).

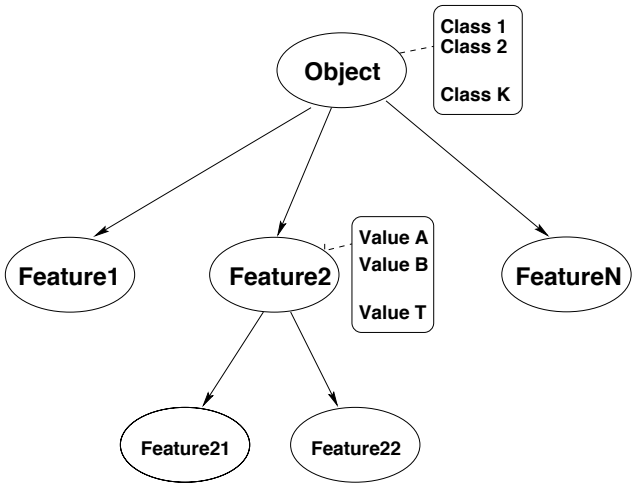

Figure 3.2. Bayesian network general structure in the Ascender II reasoning subsystem.

The feature set selected for the system can be divided into the following groups: simple features, such as width and area of a region, which can be implemented using

simple pixel counters; features that verify geometrical properties in 2D, like corners and T junctions; and 3D features, such as planar surface fit, height, and rooftop model matching, which need either a DEM (digital elevation map) or sophisticated stereo matching algorithms.

Edges were placed in the networks between features that are related, going from the most general feature to the most specific feature (like from feature 2 to feature 21 in Figure 3.2). Edges were also placed going from the root node to all nodes representing general features.

Another problem when designing Bayesian networks is to define a set of probabilities. Where do probabilities come from and how can they be determined? Features can be statistical and they can be measured over regions in different images. These measurements can be used to select the ranges for the states of the feature, and to measure frequency (i.e., the number of times feature q is in state z and the region belongs to class i divided by the number of times feature q is in state z). The frequency value can be written in terms of probability using the following expression:

$$prob(Region = Class\_i | Feature\_q = z) = \frac{\mid Regions = Class\_i \cap Feature\_q = z \mid}{\mid Feature\_q = z \mid}$$

Another way is to estimate the values of the probabilities based on personal knowledge (subjective probabilities) [44]. In this case human specialists can be used to retrieve visual information, estimating values that an ideal operator should measure. This estimate does not need to be a precise value; assuming that the features are already decomposed into states (representing ranges), only the state of the feature for each region needs to be specified. In this case the states are used as a guide and the probabilities are defined subjectively by the set of specialists.

All the conditional probability tables were defined for the Ascender II system using both approaches described above. The prior probabilities for the root nodes were defined subjectively depending on the specific data set being worked on. If nothing is

known about the distribution of regions in the data set, the prior probabilities of the classes in the root node can be specified using the uniform distribution. This approach will lead to more exploratory calls to visual operators, thus a longer processing time, but the final classification of the regions should be the same in most cases. If a priori information about the data set is known, approximate values for beliefs can be given to the classes in the root node, which will make the reasoning process more efficient.

The approach described above was used to design a set of networks (experiments using these networks can be found in [35, 36, 48], and will be described in Chapter 4). One of the problems discovered early in initial experiments resulted from setting the value of the feature directly into the feature node, which assumes that the operators are 100% reliable. This is certainly not true, especially for the values close to a threshold (boundary between two states in a node). Because of this, a node corresponding to the operator for each feature being measured in the image was added to the networks. The conditional probability table between the feature and the operator which computes the feature represents the reliability of the operator. The values of the conditional probabilities can be computed easily if the ground truth for the data sets is available; otherwise they can be estimated from a set of several runs. The general structure of the modified networks is presented in Figure 3.3, and experiments using these networks can be found in [47], and will be described in Chapters 5 and 6.

3.3 Datasets

The knowledge base in the Ascender II system was designed to allow variations in the imagery used. These variations are related to number of views available, different resolutions, different number of objects in each class, etc. For the experiments performed here the following data sets were used:

The Fort Hood dataset: 7 views with known camera parameters, varying image resolution (ground sample distance per pixel - gsd), and a corresponding DEM

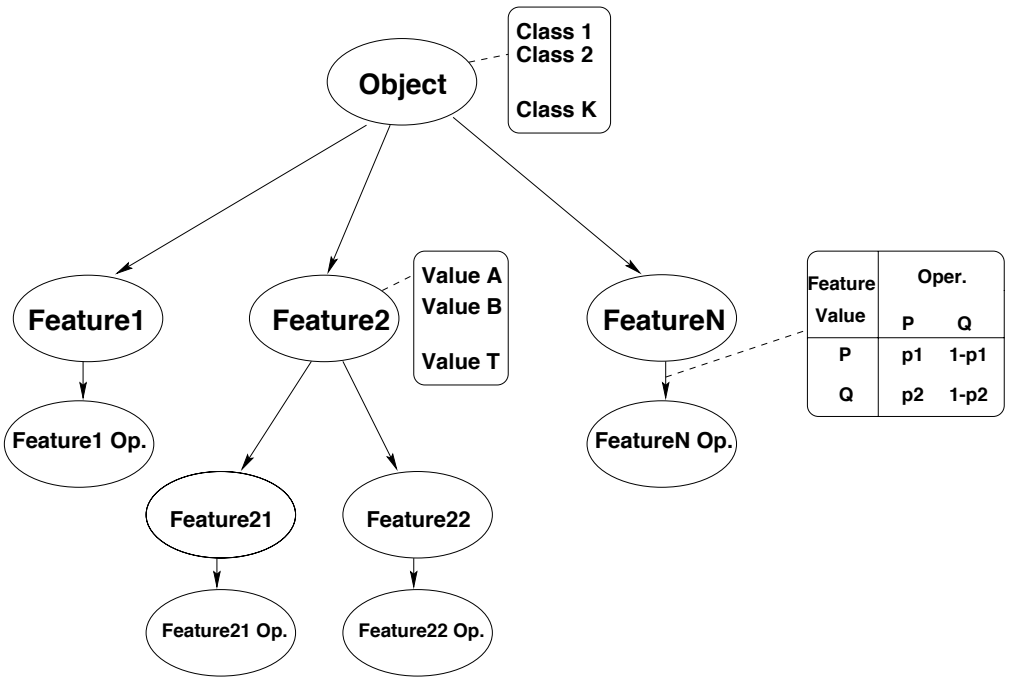

Figure 3.3. The new general structure for the Bayes Networks in the reasoning subsystem for Ascender II.

(computed using TERREST [56]). The area used for the experiments on this data set is shown in Figure 3.4. For this image the gsd value is 0.35m.

The ISPRS Avenches [1] [dataset:] 4 views, 0.1m gsd resolution, and a corresponding DEM with 0.25 m gsd resolution (the DEM was available in the dataset and not computed with TERREST). Image 5883 from this data set was used in our experiments and it is shown in Figure 3.5.

The Fort Benning dataset: 2 views, 0.14m gsd resolution, and a corresponding DEM, computed from TERREST, with the same resolution. The area called MOUT, shown in Figure 3.6, was used in our experiments.

[1] International Society for Photogrammetry and Remote Sensing; the data was obtained from the site: ftp.vislist.com/IMAGERY/PTERRAIN

Figure 3.4. Area used for experiments in the Fort Hood data set.

Figure 3.5. Image 5883 from the Avenches data set.

Figure 3.6. The MOUT site in the Fort Benning data set.

The ISPRS Flat Scene [2] [dataset:] 2 views, 0.24m gsd resolution, and a corresponding DEM. The DEM provided with the dataset has a resolution of 1 m; this resolution was considered too low for our experiments. TERREST was used to compute a DEM with a higher resolution (0.24m). The Flat Scene is shown in Figure 3.7.

The ISPRS Glandorf data set: 2 colored views, 0.24 m gsd resolution, and a corresponding DEM. Again the resolution of the DEM was too low (1 m) and TERREST was used to obtain a higher resolution DEM (0.24 m). The Glandorf area is shown in Figure 3.8.

The first three data sets (Fort Hood, Avenches and Fort Benning) were used for all experiments performed in this research. The last two data sets (Flat Scene and

[2]This dataset and the Glandorf dataset were obtained from: ftp.ifp.uni-stuttgart.de/pub/wg3

Figure 3.7. The left view of the Flat Scene data set.

Figure 3.8. The Red band of the left view of the Glandorf data set.

Glandorf) were used only as a test set for the learning experiments performed in this work. This topic will be discussed in Chapter 6.

CHAPTER 4

CONTROLLING AND MAKING DECISIONS IN AN AERIAL IMAGE INTERPRETATION SYSTEM - UNCERTAINTY DISTANCE

A human decision has both logical and subjective components of reasoning. Typically, a decision depends on three issues: the information available at the time when the decision was made, the previous knowledge of the person who made the decision about the issues involved in the decision (the logical parts), plus some other personal aspects (e.g. preferences) of the person who made the decision (the subjective part). This last statement acknowledges the fact that two people with the same knowledge about the problem, and with the same information available, might make different decisions.

Any computer system needs to make a large number of decisions during processing. The Ascender II system has three important decisions to make: what feature to select, when processing should stop, and what label should be given to a region. These decisions are directly related to the system's performance.

4.1 Feature Selection

A selective perception system is intended to solve a specified task with minimum effort [8]. For the Ascender II system, at first, a region can be any of the objects modeled in the Bayesian network, and operators are designed to measure features in the image such that an object class can be assigned to the region. It is important in selecting the operators to consider availability, cost in terms of both resources required and time for processing, reliability, etc. Thus a primary goal of the reasoning

subsystem is to select a subset of features that will allow a consistent, cost-effective identification of the region class.

The selection can be made in different ways, e.g., using a pre-defined static strategy where features are called independently of the evidence acquired, or using a dynamic strategy where different features are called depending on the current state of processing. Even if a dynamic strategy is used, there are many possible alternatives for selecting a feature. For example, the network can be simulated and the feature with the highest impact on the root node can be selected (impact here can be measured as the largest change in the belief distribution of the root node); or utility theory can be used, as in the TEA1 system [53].

The Ascender II system initially used a simple dynamic strategy for the selection process, where the node that had the highest uncertainty was selected. A new measure for uncertainty was used in this process. A random variable (N) is defined to have maximum uncertainty when the probability distribution over the states (S_N) is uniform. A measure for uncertainty, called the uncertainty distance, was defined; the uncertainty distance represents the difference between the value of the current maximum belief in the node and the value of the belief if that node has a uniform distribution (see Figure 4.1). This measure is computed as shown below (S_N represents the number of states in node N).

$$Uncertainty\ Distance\ = max(Belief(N)) - \frac{1}{S_N}$$

A more classical approach for measuring uncertainty, such as entropy, can also be used. A simple mathematical analysis comparing uncertainty distance and entropy was performed. Assume that a random variable has only two states *Yes* and *No*. The values for beliefs in this Boolean variable were adjusted as follows:

- Initially set the value of *Yes = 0* and *No = 1*.

Figure 4.1. The graph shows the uncertainty distance measure for a node with 5 states. The dotted line shows the Uniform Distribution and the state with the highest belief, which implies the largest uncertainty distance, is state 4.

- Increment the value of *Yes* and decrement the value of *No* continuously, under the constraint that *Yes + No = 1*.
- For each step measure entropy and uncertainty distance.

The curves obtained for the *Yes* value using both measures are shown in Figure 4.2. The dotted line is the uncertainty distance and the continuous line is the entropy. Figure 4.2 shows that both are symmetric and inversely related; while the entropy decreases (meaning that the uncertainty is reduced) the uncertainty distance increases (meaning that the states of the variable are moving away from the uniform distribution, thus also reducing uncertainty about the variable outcome). Experiments have shown that the Ascender II system performed slightly better when using uncertainty distance instead of entropy to select features. Because we believe it is more intuitively understandable, the uncertainty distance measure was initially used in the Ascender II system.

Given a network, the system computes the uncertainty distance for all nodes which have a corresponding IU process and selects the node with the highest uncertainty i.e. the minimum uncertainty distance. As an example, consider the case in which

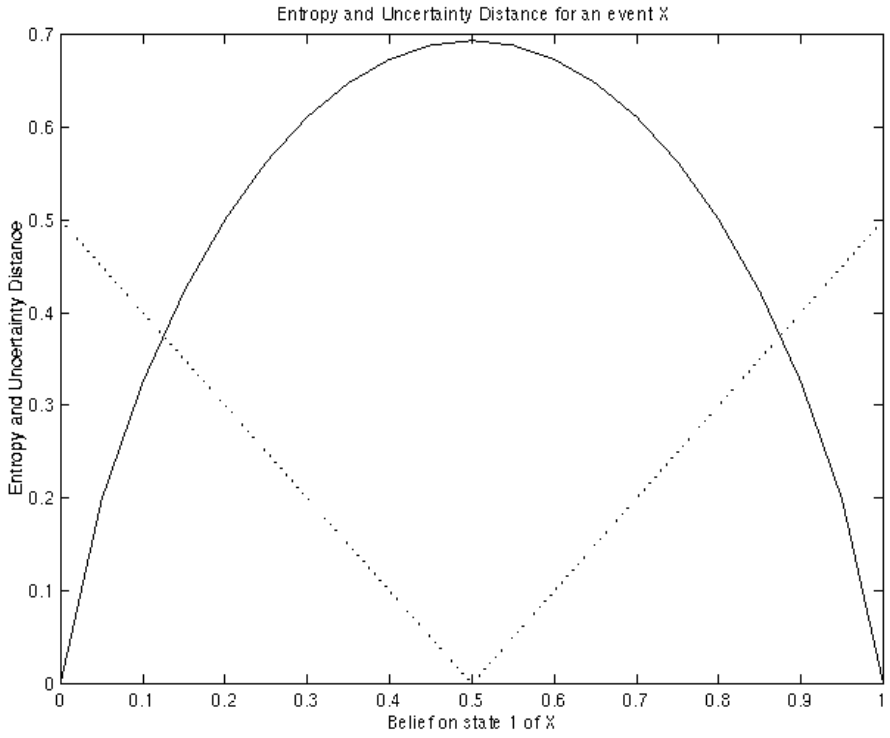

Figure 4.2. The graph shows entropy (solid line) and uncertainty distance (dotted line) for the state Yes in a Boolean variable (see discussion in text).

the features planar fit and line density inside the region are both Boolean variables. If the belief for a good planar fit in a region is 54% and the belief for a high number of lines inside the region is 67%, the uncertainty distance is higher for the planar fit variable, and this node will be selected.

Once a node is selected in the Bayesian network, a knowledge source is activated in the visual subsystem and performs an action on the image. The findings of this action are then returned to the reasoning subsystem, entered as evidence and propagated through the network. The process is repeated until the reasoning subsystem has enough evidence to recognize and label the region as one of the available object classes.

4.2 The Recognition Process

The recognition process in the Ascender II system could be defined in a number of different ways. One of the simplest recognition processes is just to define a fixed threshold. When a belief value in the root node reaches the threshold, the system stops the recognition process. In this case the region is labeled using the class which passed the threshold in the root node. The threshold could be a single global value accross all networks in the system, however, this method has some problems. If the number of classes in the root node is different for each network, it will be easier to reach the threshold in nodes with a smaller number of states than in nodes with a larger number of states. Another problem with this method is that if the threshold is set to a high value, this value might not be reachable in some networks, which implies that all applicable vision routines will be executed without being able to reach a decision.

Another approach for defining the recognition process is to use a relative threshold. The relative threshold can be measured either within the state or between states. In the first case it is measured by the relative amount (percentage) of how much the

belief in each state changes, in one step of the reasoning process, or in P steps; if the change reaches a certain threshold the process stops and the region is labeled using the class which has passed the specified threshold. In the second case, the two largest beliefs in the root node are compared, and the reasoning process is stopped if one of the beliefs is at least a factor k times larger than the other. Again, once the reasoning process is stopped, the region is labeled using the class with the highest belief in the root node.

The decision criterion first used to select a label for a region was a relative threshold, where the two classes with the highest and second highest beliefs are compared after each new piece of evidence is propagated through the network. If the maximum belief is at least k times the value of the second highest belief the controller stops and identifies the region as belonging to the class with the maximum belief.

Clearly, the value of k has an impact in the system's performance. The value of k is directly related to the correct classification of a region, the number of operators required to classify a region, and the number of times that the system can not decide about the region using the criterion but instead has to make its current "best guess" (label the region using the object class with the highest belief in the root node). If k is too small the system might not apply any available operator and take the current prior beliefs as acceptable, which will lead to a poor classification process. If the value of k is too large the system would use all operators available and still not reach the decision criterion. In this case it will have to make the "best guess" about the region's class. The system inspects the beliefs in the root node and finds the highest value among them, and labels the region using the object class with the highest belief in the root node. In order to set a value for k a set of experiments were defined. In these experiments the value of k was changed from 1.25 up to 6.0 and the number of regions correctly classified (see Figure 4.4), the number of operators used (see Figure 4.3), and the number of times the system was not able to decide using the criterion (see

Figure 4.5) were observed. For this experiment the Fort Hood, Avenches, and Fort Benning datasets were used, with a total of 79 regions and 906 possible operators.

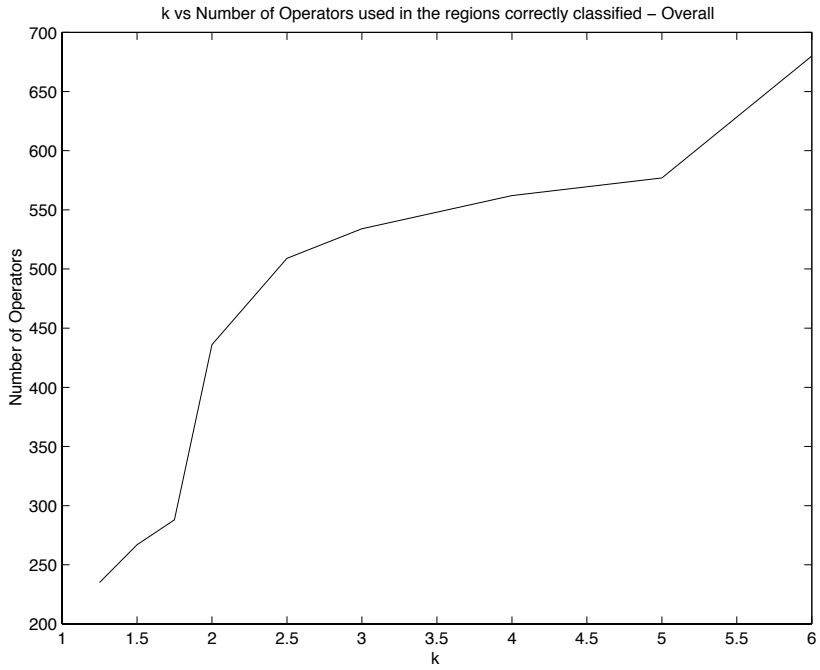

Figure 4.3. Number of operators vs. threshold k: the graph shows the total number of operators used by the system when the value of k goes from 1.25 to 6. If the system used all evidence available, prior to a decision, it would use 906 operators.

As expected, Figure 4.3 shows that the number of operators required to reach the decision criterion increases when the value of k increases. The number of regions correctly classified also increases when k increases (Figure 4.4). This was expected as a higher value of k requires a larger number of operators applied to the region, resulting in a more reliable classification process.

Some visual operators have a set of parameters that can be adjusted (or calibrated) to get a better performance based on the image conditions (brigthness, contrast, noise, etc). The parameters for the visual operators used in these experiments were manually set so that they would perform well on the three datasets used. If the features retrieved by the operators for each region were correct and reliable the graph

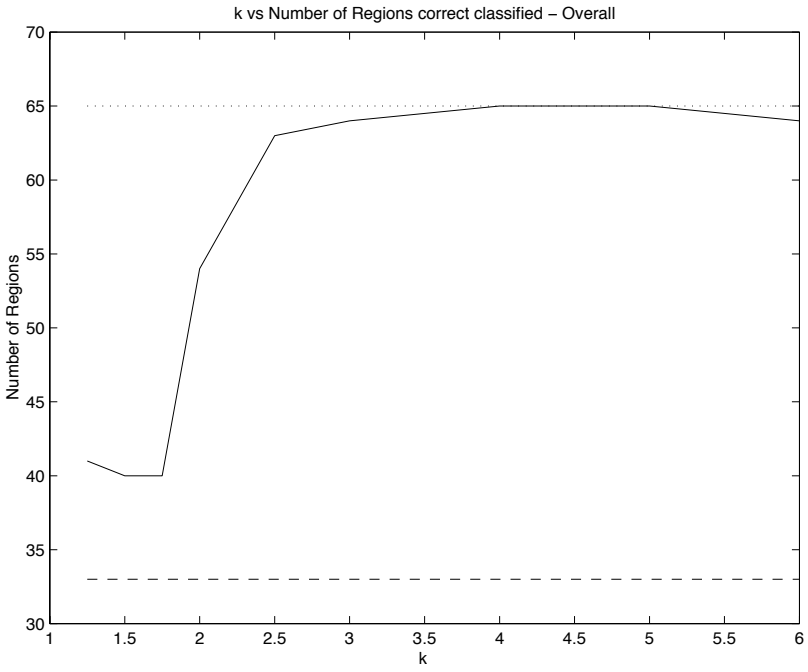

Figure 4.4. Regions correctly classified vs. threshold k: the graph shows the number of regions correctly classified when the value of k goes from 1.25 to 6. The dotted line shows the number of regions correctly classified if the system uses all available operators, and the dashed line shows the number of regions correctly classified using only the initial prior probabilities.

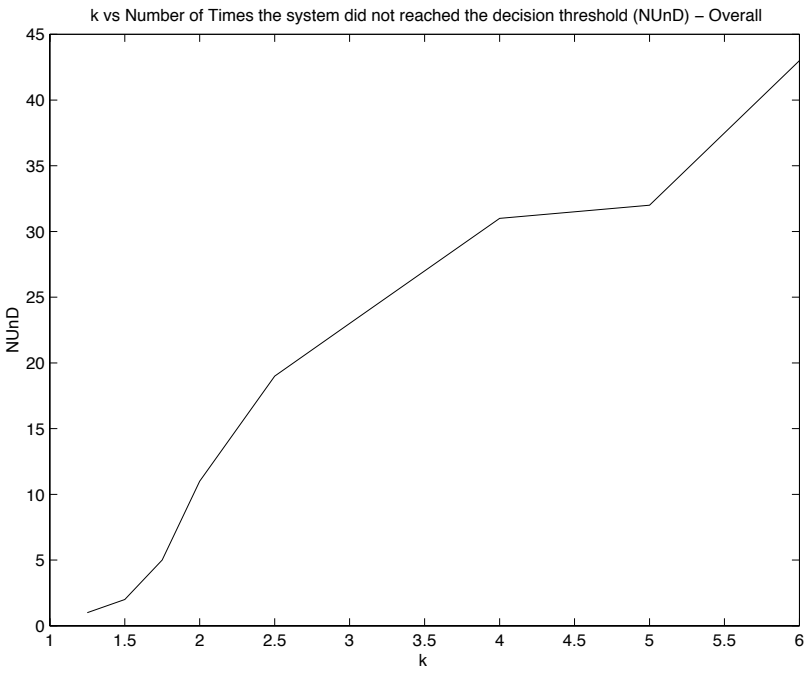

Figure 4.5. Number of times the system did not decide based on the threshold k criterion: the graph shows the number of times (out of a total of 79 regions) the system did not reach the decision criterion when the value of k goes from 1.25 to 6. In these cases the system made the best possible guess about the region's class.

in Figure 4.4 should be strictly monotonic. As the image set used is noisy and does not correspond to exact conditions where the operators were calibrated, the outcome of the operators, when applied to this images, might not correspond to the real feature value, which might lead to misclassifications. Finally, as shown in Figure 4.5, the higher the value of k the larger the number of regions for which the system could not reach the decision criterion. The value of k has to be defined based on these three issues. A value of k equal to 2 or 2.5 would give the best overall performance. For our experiments the value 2 was used.

4.3 System Description

The first set of experimental networks are presented in Figures 4.6, 4.7, 4.8, and 4.9. At this point only buildings are handled by the system beyond the level 0 network. The network at level 0 (Figure 4.6) attempts to recognize the class to which the region belongs. If the class identified is not *Building* the process is stopped, and a generic 3D model for the object class is selected from the data base for visualization purposes. In the case where a building is identified the network at level 1 is called (Figure 4.7). This network is designed to identify single-level buildings based on a simple set of features; if a single-level building is identified the network shown in Figure 4.8 is called to determine the rooftop class. If a multilevel building is identified, there is a possibility that the hypothesis is wrong (a multilevel building may have a line on the roof which looks very much like the center line of a peaked roof). Consequently, the network shown in Figure 4.9 is called to confirm a multilevel building, or to backtrack to a single-level building (this is an example of a pre defined backtracking path encoded into the system. If the building is not a multilevel building the system makes a call to the Rooftop network shown in Figure 4.8). If a multilevel building is identified, the system breaks the region into two new subregions, based on the evidence gathered, and calls the network at level 1 for each subregion recursively.

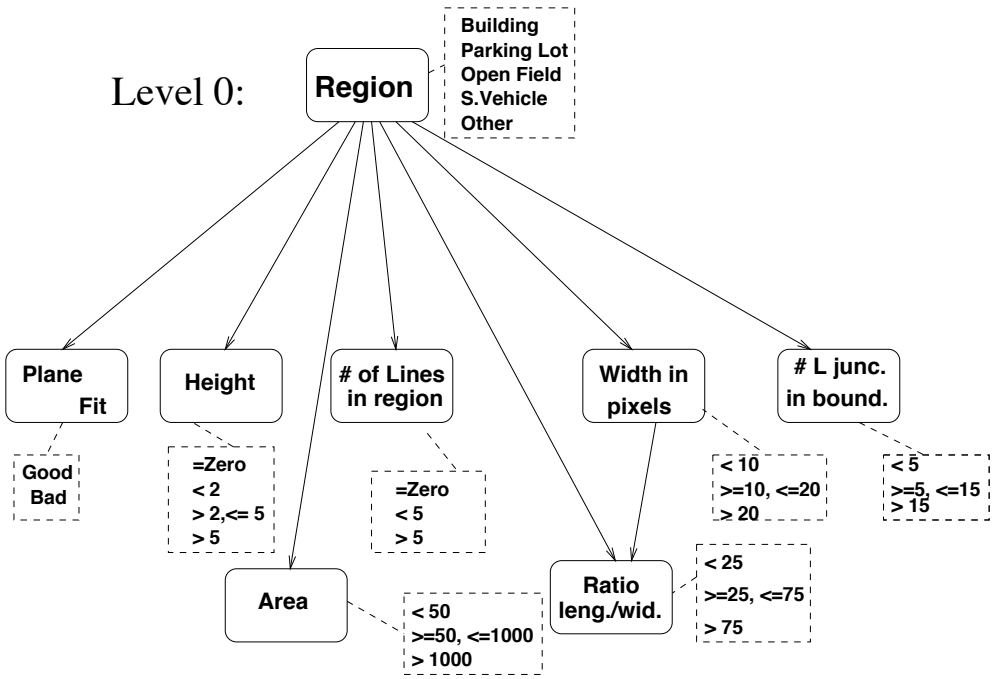

Figure 4.6. The level 0 network determines if a region belongs to one of the possible object classes: Building, Parking Lot, Open Field, Single Vehicle, and Other.

Note that some knowledge sources can be called from different levels. Each call is related to a specific region and values already computed are stored in the visual subsystem. When a repeated call is made, the system will return the value stored in the database and will not recompute the feature.

4.3.1 How the Ascender II System Works - Snapshots

This section presents a sequence of snapshots taken when the system was running over a region in the Fort Benning data set. The input image and the regions to be identified are shown in Figure 4.12; this set of regions was determined using optical images processed by the Ascender I system (Figure 4.10) combined with a SAR classification provided by Vexcel Corp (Figure 4.11). The prior probabilities associated with the object classes in the level 0 network are shown in Table 4.1.

Consider the sequence of events which occurs when the system is processing region X in Figure 4.12. This region is shown in Figure 4.13 (top) with the network at level

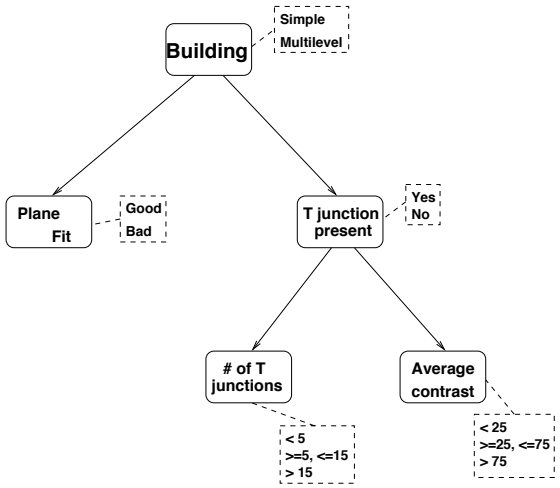

Figure 4.7. The level 1 network is invoked for each building found in level 0 to determine if it is a single-level building or a multilevel building.

0. The nodes shaded are the ones invoked by the reasoning subsystem; the evidence found during inference and the class identified are shown in the figure. After deciding on a building class the system called the network at level 1, which checked for T junctions in focus-of-attention terminal areas shown in Figure 4.13 (bottom). The terminal areas are defined automatically and designed to have a length of 50% of the length of the region and have a width of 10% the width of the region. Evidence from level 1 identified the building as multilevel and the network to confirm multilevel building at level 2 was invoked.

At this point the system decomposed the region into two new subregions, as shown in Figure 4.14 (top), and recursively called the network at level 1 for each new subregion. The process is repeated on each of the two subregions, as shown in Figure 4.14 (bottom). Region A was also identified as a multilevel building and decomposed into two new subregions.

The process is again repeated for the left-most subregion (A1). This region was identified as a single-level building, so the rooftop network is called for this particular region in an attempt to identify the building's rooftop. In this case, the system identifies the rooftop as a peaked roof (see Figure 4.15 top). The process is repeated

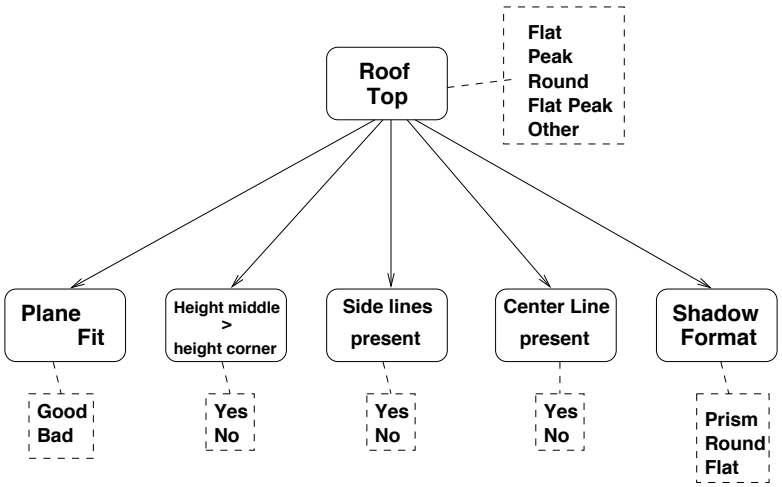

Figure 4.8. This network is invoked at level 2. It is called after a single-level building is detected and it is used to determine the rooftop type.

Table 4.1. Probability distribution of beliefs in the root node for the level 0 network among the states Building, Parking Lot, Open Field, Single Vehicle and Other for all three data sets.

Knowledge	Building	Parking Lot	Open Field	Single Vehicle	Other
Fort Hood	0.35	0.2	0.2	0.15	0.1
Avenches	0.32	0.25	0.38	0.03	0.02
Fort Benning	0.35	0.2	0.2	0.15	0.1

for subregion A2, which is also identified as a peaked roof building. The system then considers region B (Figure 4.15 top) and eventually identifies two peaked roof structures, as shown in Figure 4.15 (bottom), which is representative of the final result.

4.4 Experiments and Results

In all experiments described here, only the domain-specific knowledge in the network at level 0 was changed from one experiment to the other. This knowledge represents expected frequency for each possible class in the root node and it is represented as prior probabilities (see Table 4.1).

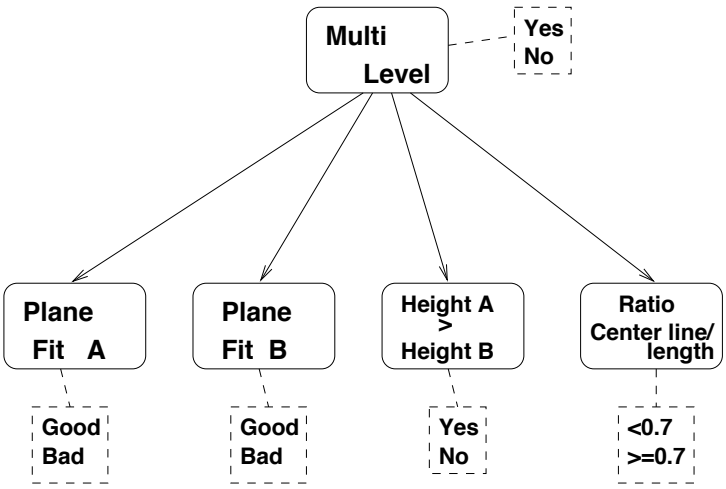

Figure 4.9. This network is also invoked at level 2. It is called when a multi-level building is detected. The reasoning system calls this network to confirm the recognition at level 1.

Table 4.2. Average number of calls to knowledge sources in the level 0 network, for different data sets, for all classes (Total column), and by specific classes (remaining columns).

Knowledge	Total	Build.	P.Lot	O.Field	S.Vehicle
Fort Hood	3.9	4.25	4.00	3.29	0
Avenches	4.7	5.22	5.00	4.0	0
Fort Benning	2.9	3.00	0	0	2.0

There are seven visual operators available in the network at level 0 and three additional operators at level 1. Table 4.2 presents the average number of visual operators invoked by the reasoning subsystem at the level 0 network for each data set, and also the average calls by region type (object class). The recognition results obtained were first presented in [48], and they are summarized in Table 4.3.

Notice that in all cases misclassification was decided relative to the initial focus-of-attention regions provided to the system and not against a complete ground truth model. That is, if a building was not included in the set of focus-of-attention regions, and consequently not recognized by Ascender II, it was not counted as an error.

The recognition results obtained for the regions (Fort Hood data) shown in Figure 4.16 are shown in Figure 4.18. The undecided region (region A) has a belief of 59% for

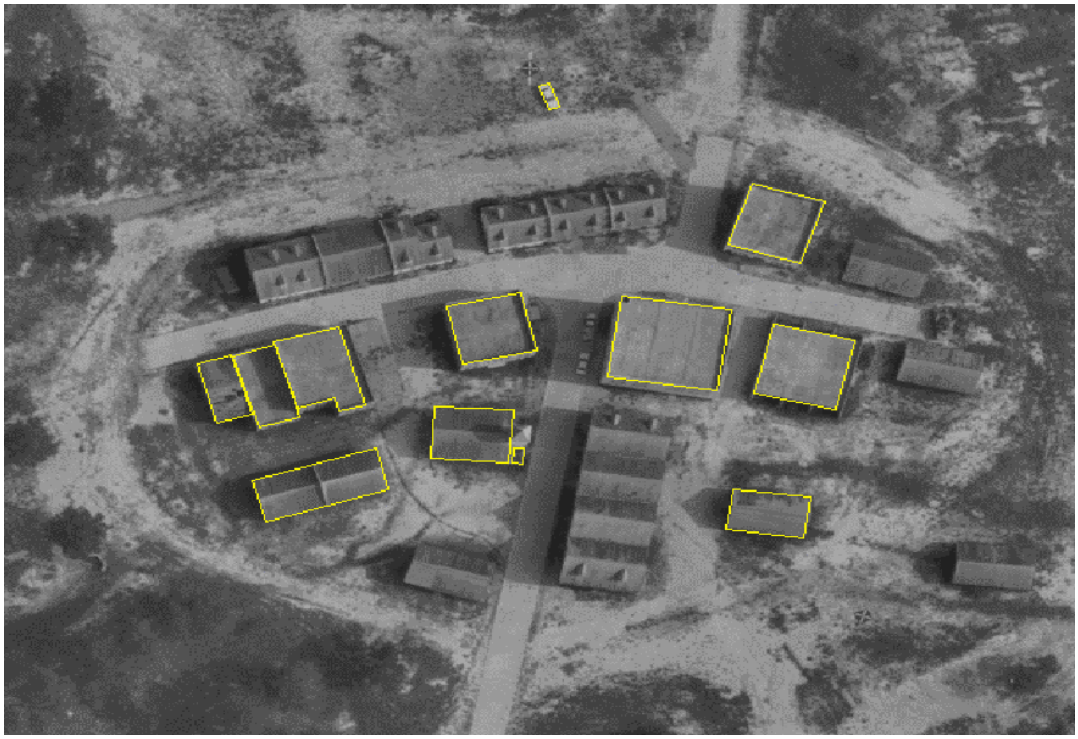

Figure 4.10. Regions obtained by the Ascender I system from the Fort Benning dataset.

Table 4.3. Summary of the recognition process for different data sets. In each case the number of objects correctly identified is shown, followed by the total number of objects evaluated by the system (* - see text concerning identity of the RV regions).

Data set	Overall	Level 0	Level 1	Level 2
Fort Hood	37/41	37/41	21/21	21/21
Avenches	17/18*	18/18	16/16	15/16
Fort Benning	17/19	18/19	17/17	16/17

Figure 4.11. Regions obtained from SAR data from the Vexel Corp.

Figure 4.12. Regions to be identified by the Ascender II system from Fort Benning dataset. The region marked with an X was selected to demonstrate the performance of the system. See text for a description of how the regions were obtained.

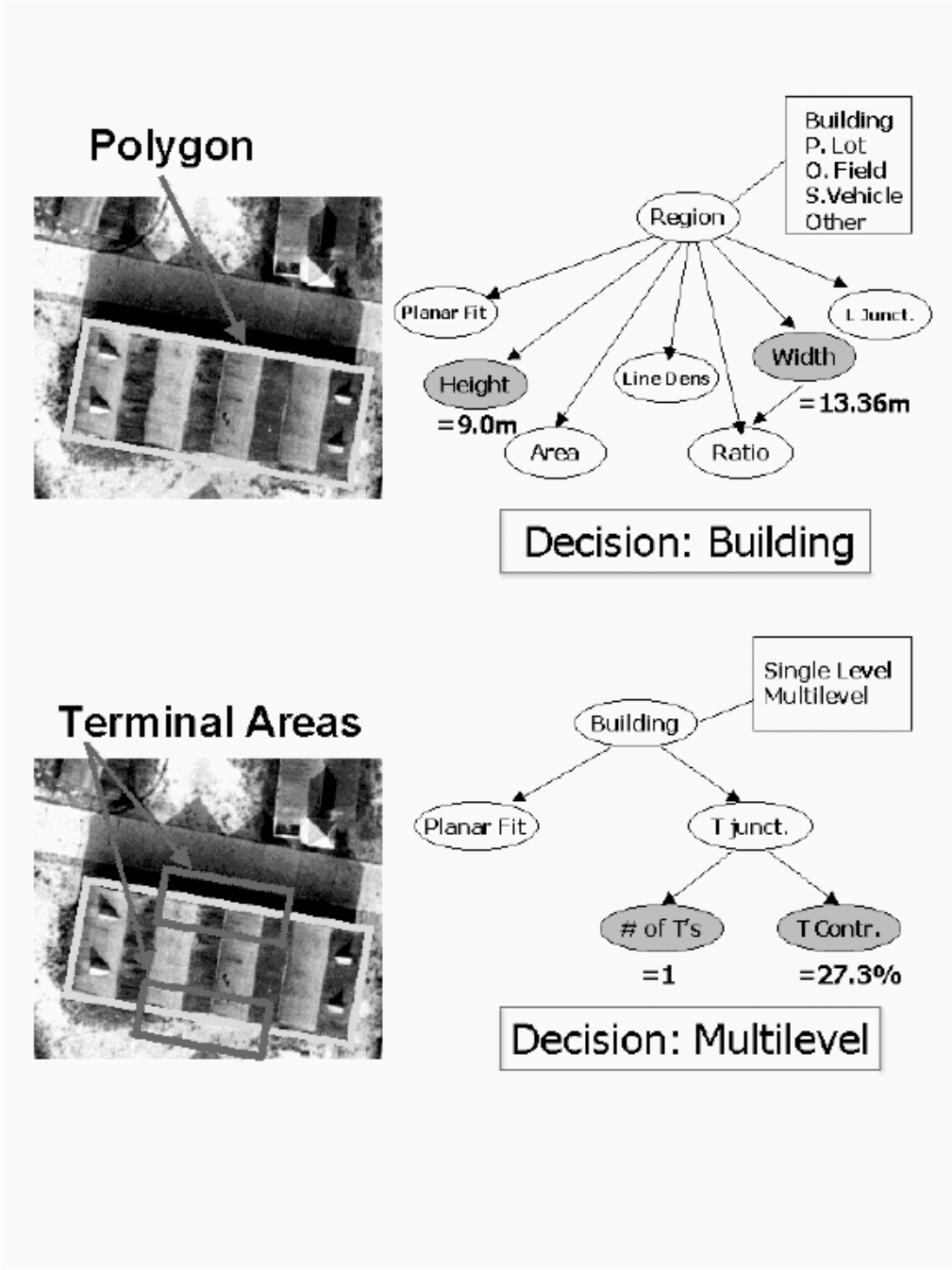

Figure 4.13. Decomposition of the building hypothesis. Top: Region of discourse. The ground truth is a multilevel building with 4 peaked roofs. The building class was selected as the most probable because of the combination of the features Width = 12.36 m and Height = 9.0 m. Bottom: The system used the fact that the region was identified as a building and that evidence of T junctions was found (Number and Contrast of T's) to identify it as a multilevel building.

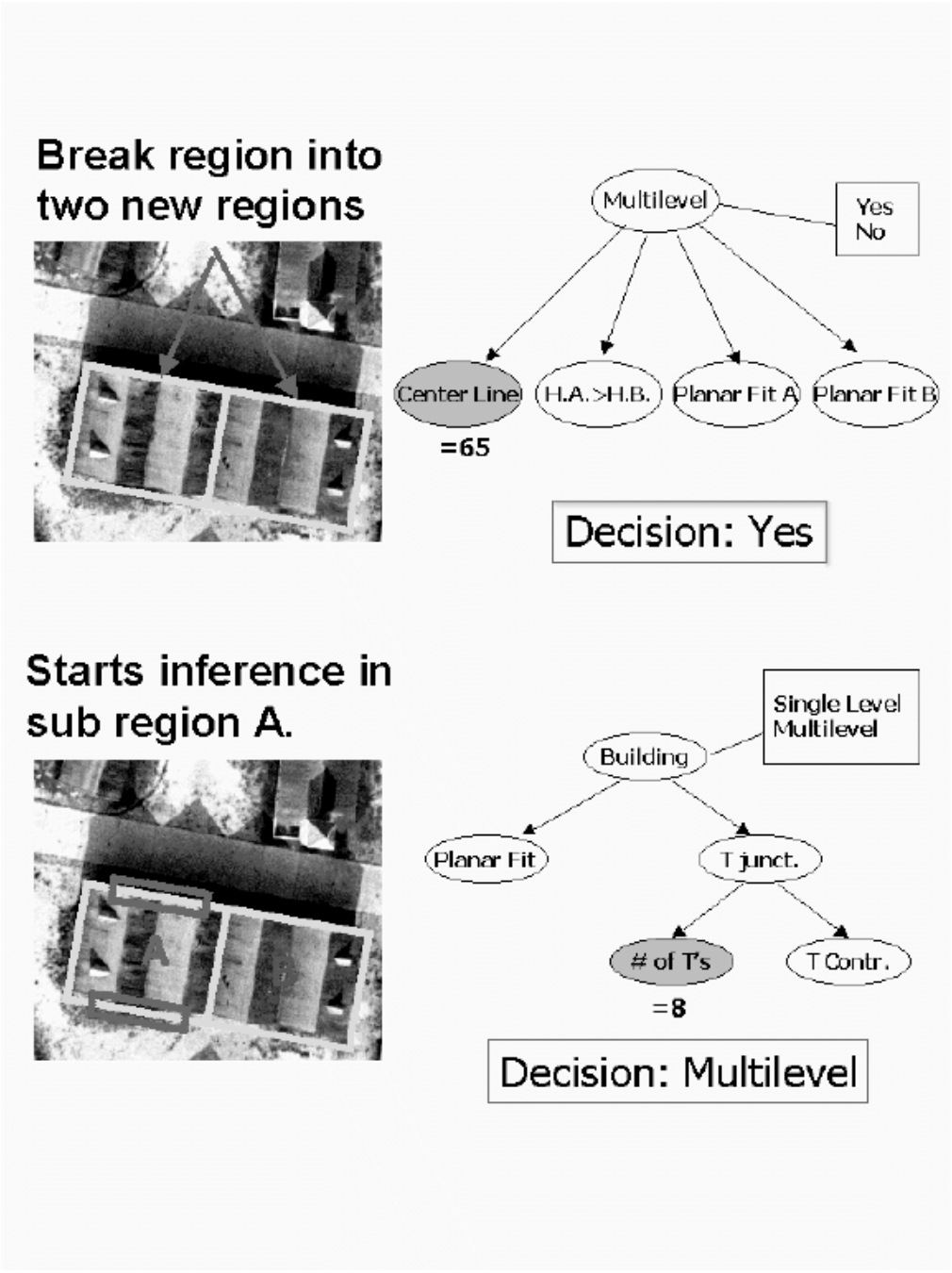

Figure 4.14. Decomposition of the building hypothesis. Top: The region was confirmed as a multilevel building and was broken into two new subregions. Bottom: Looking for T junction evidence in the terminal areas of region A.

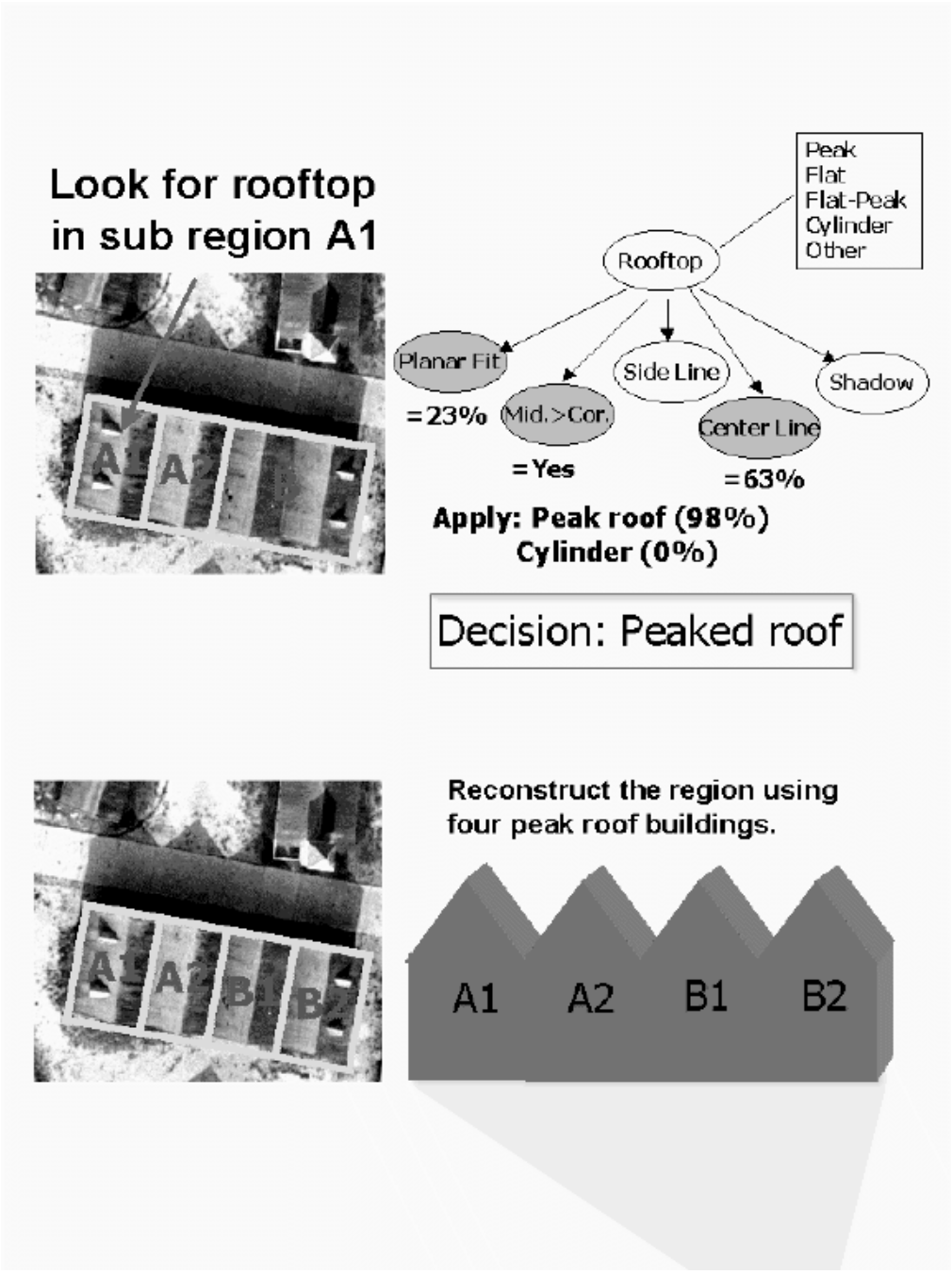

Figure 4.15. Decomposition of the building hypothesis. Top: Identified the region A1 as a peak roof building. Bottom: Reconstructed 3D model for the whole region.

Figure 4.16. The input regions from the Fort Hood data set. These regions were obtained by running the original Ascender I system constrained to detect two-dimensional building footprints [16].

the Parking Lot state and 31% for the Open Field state (the region is a parking lot). By the decision criterion discussed earlier (section 4.1), although close to the ratio, no decision is possible (this does not happen with our utility theory mechanisms, see Chapter 5.). The misclassified regions are: an open field identified as building (region C), and 2 single vehicles identified as open fields (regions F).

The classification obtained for the regions in the Avenches site (Figure 4.17) is presented in Figure 4.19. There is a set of small regions in the Avenches data set that were detected by the Ascender I system as buildings, and confirmed by the Ascender II system as single buildings with flat roofs; the correct identity of these objects is unknown. They look like large RV's or large containers, as one can see by their shadows and relative size. We considered these areas as large RV's which could be identified as either flat roof buildings, or single vehicles. The system's knowledge base

Figure 4.17. The input regions from the Avenches data set. These regions were obtained by running the original Ascender I system constrained to detect two-dimensional building footprints.

was re-engineered to allow this type of discrimination (see Chapter 5). In this case the system correctly identified them as flat roof buildings. Region A is a parking lot for boats and it was correctly identified as parking lot. The rooftop of building B was misclassified as a peak instead of flat.

The results for the Fort Benning data are presented in Figure 4.20. The only complete misclassification was a small peaked roof building (which is the church steeple), in front of the church, which was classified as a single vehicle. The other problem found was a flat roof building classified as peak roof building due to a shadow that generates strong evidence for a center line in that building (see Figure 4.21).

The final 3D reconstruction for the Fort Hood and Fort Benning data are presented in Figures 4.22 and 4.23 from a synthesized or virtual viewpoint. Geometric reconstruction of building models is based on a robust surface fit to the local DEM us-

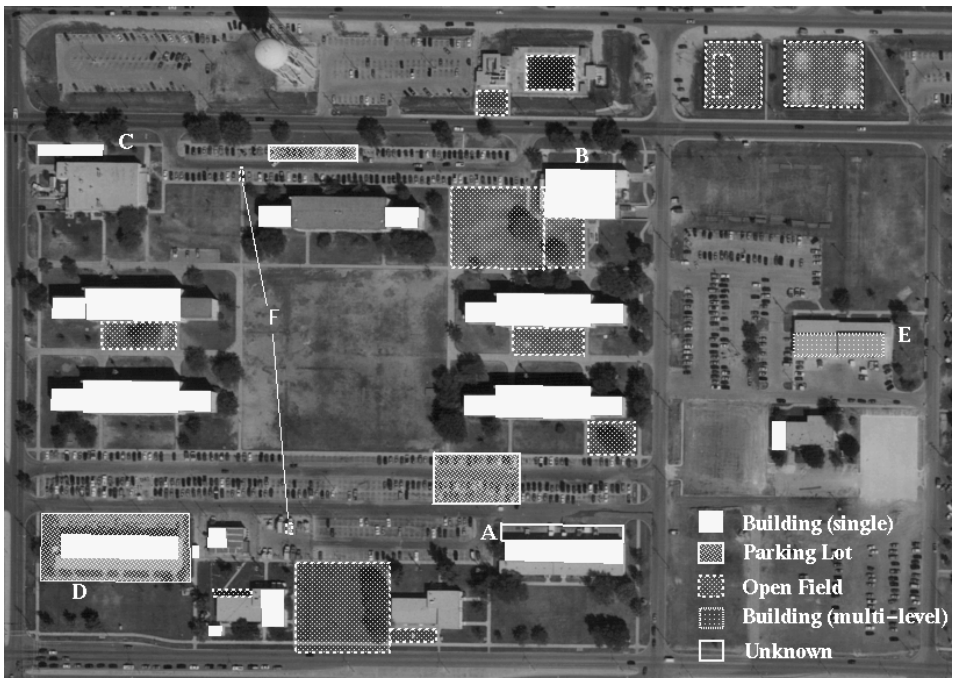

Figure 4.18. Recognition results on the Fort Hood data. Three regions were misclassified and for one region the system was not able to make a decision.

Table 4.4. Mean, maximum and minimum errors, in meters, for the 3D reconstruction in Fort Benning.

	IV Planimetric	IV Altimetric	IV Absolute
Mean	0.528	0.608	0.805
Maximum	0.848	0.892	1.112
Minimum	0.246	0.377	0.524

ing the initial model and parameters generated by the recognition process [34]. Error analysis was performed by comparing the 3D reconstruction obtained with hand-constructed CAD models available for the Fort Benning data. The errors obtained in this analysis are shown in Table 4.4 in Planimetric (horizontal), Altimetric (vertical) and Absolute distances, all in meters.The maximum error was 1.1 meter (or about 10 pixeis) and, given the quality of the images for this site and the quality of the camara parameters to compute the DEM, this result can be considered good.

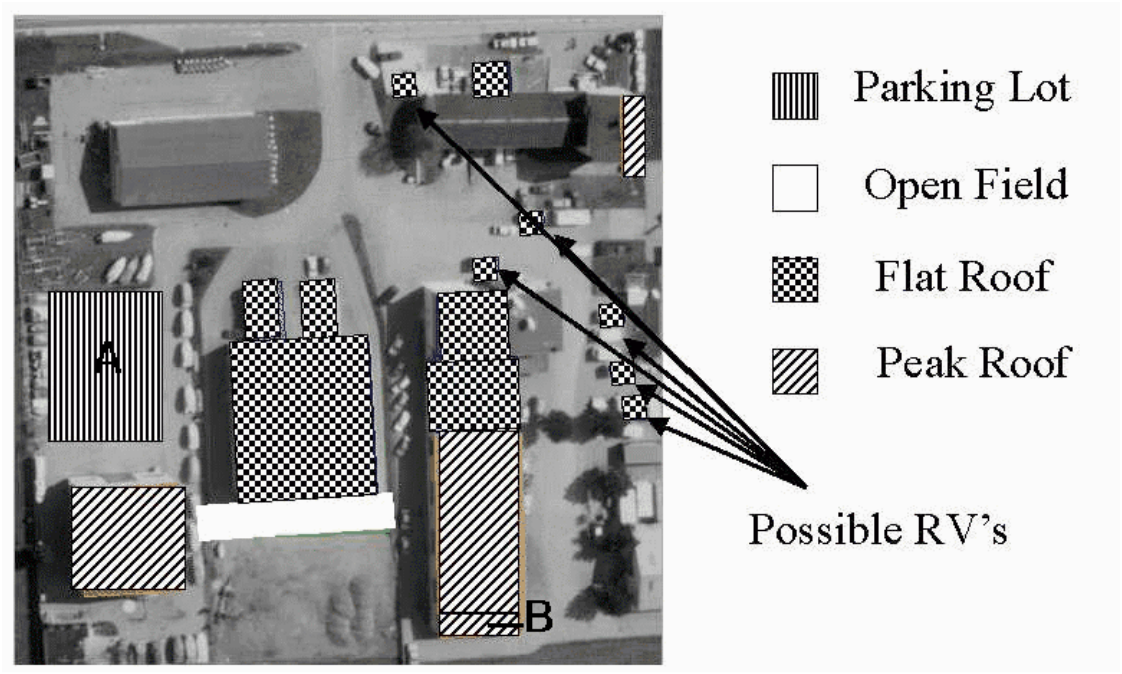

Figure 4.19. Recognition results on the Avenches data set. All R.V.'s were classified as flat roof buildings.

4.4.1 Evaluation

One of our claims is that the Ascender II system can perform 3D reconstruction accurately and efficiently. In order to validate this claim the Fort Benning data was used to test the Ascender II system against a system that randomly selects a vision operator, and also against a system that invokes all operators to get all available evidence prior to making a decision. The summary of these results is presented in Table 4.5. For the system with random selection only the best performance is shown. Table 4.6 shows the average number of operators called using each method in the evaluation process. For the systems with the best performance, Ascender II and All Evidence, the time required to process each region, on average, is shown in table 4.7.

The main difference, in terms of identification, between the Ascender II system and the All Evidence system was that the small building in front of the church was classified as single vehicle by the Ascender II system. The system using all available

Table 4.5. Number of regions correctly identified and total number of regions at each level for the Fort Benning data set.

Method	Overall	Level 0	Level 1	Level 2
Random	15/19	16/19	15/15	14/15
All Evidence	17/19	19/19	17/18	16/17
Ascender II	17/19	18/19	17/17	16/17

Table 4.6. Average number of operators called for each region for the Fort Benning data set.

Method	Level 0	Level 1	Level 2
Random	3.04	1.31	2.84
All Evidence	7	5	5
Ascender II	2	1.05	2

Table 4.7. Average processing time (in seconds) on each region for Ascender II and All Evidence methods for the Fort Benning data.

Method	Level 0	Level 1	Level 2
All Evidence	39.6	24.8	1.68
Ascender II	11.3	24.6	0.89

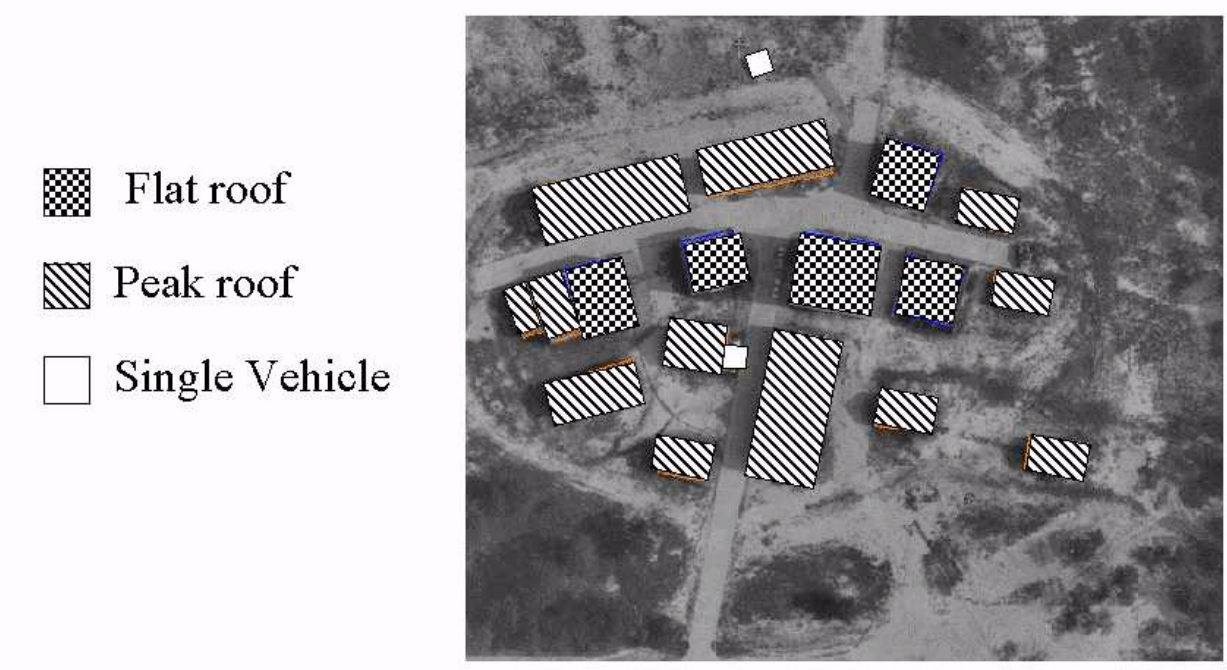

Figure 4.20. Recognition results on the Fort Benning data set.

evidence correctly identified it as a building, but misclassified its rooftop as a cylinder and not a peak. The price paid by the All Evidence system was much higher in terms of time required for the correct classification. At the level 1 network the difference is higher in terms of operators than in terms of time. This network has only three operators and two of them are applied twice (once for each terminal area). The planar fit operator is a very expensive operator in terms of time, and it was always called by the Ascender II system. This made the difference in terms of time less relevant.

The random system generated more misclassifications, exchanging buildings for parking lots and open fields, as shown in Table 4.8. In the best case, three regions were misclassified at the level 0 network. Notice that after 5 runs using the Random operator selection, region number 1 was never classified as a Flat roof building, all regions were misclassified at least once, and none of the Random runs did better than

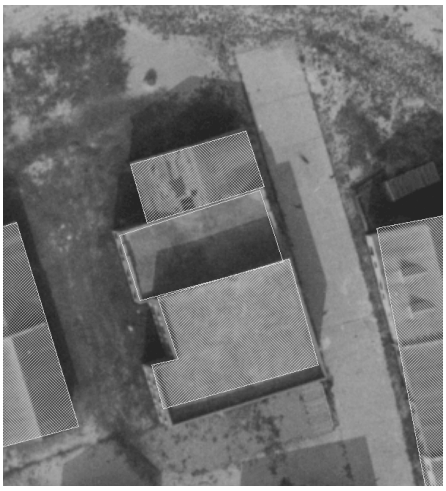

Figure 4.21. The shadow on the flat roof of this building makes a strong evidence for a center line and the roof top was misclassified as a peaked roof building.

the All evidence system or the Ascender II system . In this case the overall probability of misclassification is 0.326, a high value for a classification system.

Table 4.8. The table presents the results of the 5 random runs executed in the Fort Benning data set. The true column gives the correct classification for each region and the Run columns give the classification given for each region based on a random selection of operators. P stands for Building, Single level, Peaked roof; F is the same for a Flat roof; V stands for Single Vehicle; M for Building, Multilevel; L for Parking Lot; O for Open Field. The * indicates the misclassified regions. The errors column shows the number of times the region was misclassified in 5 runs, and the errors row shows the number of misclassifications per run.

Region ID	True	Run 1	Run 2	Run 3	Run 4	Run 5	Errors
0	P	P	P	L*	V*	P	2
1	F	P*	P*	P*	P*	V*	5
2	F	F	F	L*	F	F	1
3	P	P	V*	P	P	P	1
4	P	L*	F	F	F	F	1
5	P	L*	P	L*	P	P	2
6	BS	V*	F	F	F	P	1
7	V	P*	V	V	V	V	1
8	F	F	F	F	O*	F	1
9	F	F	P*	F	F	F	1
10	F	F	L*	L*	L*	F	2
11	P	V*	P	P	P	P	1
12	P	P	L*	L*	P	P	2
13	P	P	P	V*	P	P	1
14	P	P	F*	P	P	F*	2
15	M	M	M	M	L*	L*	2
16	P	P	V*	P	P	V*	2
17	P	P	M*	P	M*	P	2
18	P	P	L*	P	P	P	1
Errors	-	6	8	7	6	4	

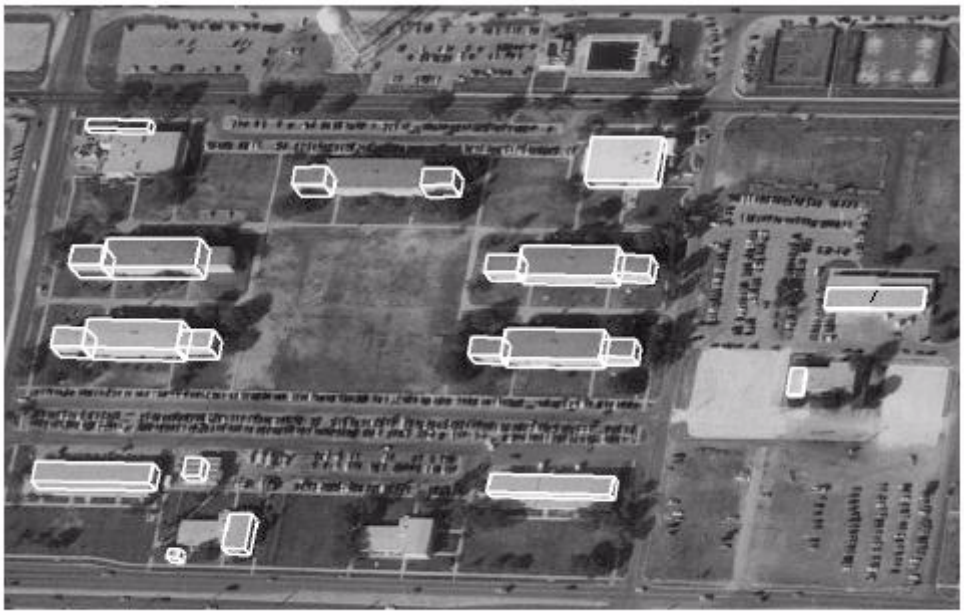

Figure 4.22. 3D reconstruction on the Fort Hood data.

Figure 4.23. 3D reconstruction on the Fort Benning data.

CHAPTER 5

CONTROLLING AND MAKING DECISIONS IN AN AERIAL IMAGE INTERPRETATION SYSTEM - UTILITY THEORY

One of the problems with uncertainty distance is that personal preferences are not easily modeled in that framework. The only way to add personal preferences using uncertainty distance is to change the conditional probability tables in the Bayesian networks. Another problem with uncertainty distance is that feature selection is based only on one state of each feature (the one with the highest uncertainty distance). Utility theory allows the modeling of personal preferences just by changing the order of consequences, which directly affect the utility tables. Utility theory uses all states in a feature node to compute the expected value of information of the feature during the selection process, and it is a well defined methodology [44] based on probability theory.

5.1 Reasoning in the Ascender II using Utility Theory

The Ascender II system has to select a set of visual operators to help it in deciding the region's label. There are K features that can be measured in the region, the measurements are not completely reliable, the measurements can have different costs, and at least some measurements are required to decide about the region's label. Utility theory is a probabilistic technique for decision making and it fits well in a Bayesian network system. Utility theory provides mechanisms to select the decision that has the highest expected utility. In the discussion that follows, we use the following notation:

- $R_j \stackrel{\text{def}}{=}$ the event that region R belongs to Class j (the outcome).
- $DR_j \stackrel{\text{def}}{=}$ the decision that region R is identified as Class j (the decision based on the outcome).
- $E \stackrel{\text{def}}{=}$ all the evidence collected so far.
- $F_m \stackrel{\text{def}}{=}$ feature F is discretized into m states.

5.1.1 Selection of a Visual Operator

The region's prior probabilities and the conditional probability tables relating features with labels are obtained as described in Chapter 3 and stored in the Bayesian networks. The utility tables storing the values $U(DR_i|R_j)$ (the utility of deciding that a region is in class i given that the region belongs to class j) are not hard to define. The tables are based on the consequences of classifying a region correctly or not, and this is related to the user's preference for a certain object class and/or specific goals for the classification process (e.g. find all parking lots in the image) [44]. The utility tables used in most of the experiments are all similar, with 1's on the diagonal and 0's in all other entries (see Table 5.1). In this case, only the correct labels are accepted and they have exactly the same utility in the classification process. Later in this chapter it will be shown how to set preferences using utility tables and, at that point, different utility tables will be presented.

The expected utility (EU) of each decision is computed using the probability that a region belongs to a class j given all knowledge available, $P(R_j|E)$, and the utility of deciding that a region is in class i given that the region belongs to class j, $U(DR_i|R_j)$, [44]:

$$EU(DR_i|E) = \sum_{j=1}^{N} U(DR_i|R_j) * P(R_j|E) \tag{5.1}$$

Table 5.1. The table shows all utilities for the level 0 network in the Ascender II system.

Decide	*Class*			
	Building	Park. Lot	Open Field	Other
Building	1	0	0	0
Parking Lot	0	1	0	0
Open Field	0	0	1	0
Other	0	0	0	1

The system's utility is defined as the maximum value among the expected utilities of each decision:

$$EU(DR_\alpha|E) = max(EU(DR_i|E)) \tag{5.2}$$

The best decision is defined as the decision α which gives the maximum expected utility:

$$\alpha = argmax_i(EU(DR_i|E)) \tag{5.3}$$

In our problem domain the system has to decide the most likely identity (e.g. label) of a region. Assume that there are K features that can be measured in the region and that the measurements are not completely reliable, as it is almost always the case. Features are selected based on their value of information [32]. The value of information gives the expected gain, in terms of utility, that is achieved by knowing the feature value. This value is computed as follows: for each feature currently available compute the expected utility of the system given that information about the feature is known, $EU(F_m)$. The feature is divided into M discrete intervals and each of these intervals might lead to a different object class (for example, if height is less than 1 meter the region would be labeled as open field, but if the height is greater than 4 meters the same region would be labeled as a building), with a corresponding

utility. The utilities found for each interval have to be weighted by the probability that the feature value is in the corresponding interval:

$$EU(DR_i|E, F_m) = \sum_{m=1}^{M} P(F_m) * max_i(EU(DR_i|E, F_m)) \tag{5.4}$$

Now, compute the value of information of each feature by subtracting the current utility of the system from the expected new utility of the system when information about the feature is known, as shown in the equation below:

$$VI(F_m) = EU(DR_{\alpha'}|E, F_m) - EU(DR_\alpha|E) \tag{5.5}$$

and select the feature with the highest value of information. Intuitively, the value of information measures the expected improvement in the utility once the result of an operator becomes available. Notice that the best decision can change (α' and α can be the same or not) once the most valuable feature is measured and its evidence is entered into the system.

Figure 5.1 shows a generic Bayesian network that will be used to illustrate how feature selection is performed in the Ascender II system. The first step is to compute the system's utility before extracting any information about the features. This value gives a reference for selecting features and helps on deciding the most valuable feature. Each decision has an expected utility $U(Dec_i) = EU(DR_i|E)$; the expected utilities of the decisions can be calculated by multiplying the matrix of utilities by the column vector of beliefs from the root node, as shown in Figure 5.1. The system's utility is the maximum value among the utilities of the decisions.

Once the system's utility is known, the next step is to determine if it is worth to measuring any feature and, if it is, what feature should be measured. In order to do this the value of information for each feature has to be computed. This is performed by computing the expected utility of each feature as follows: let us assume

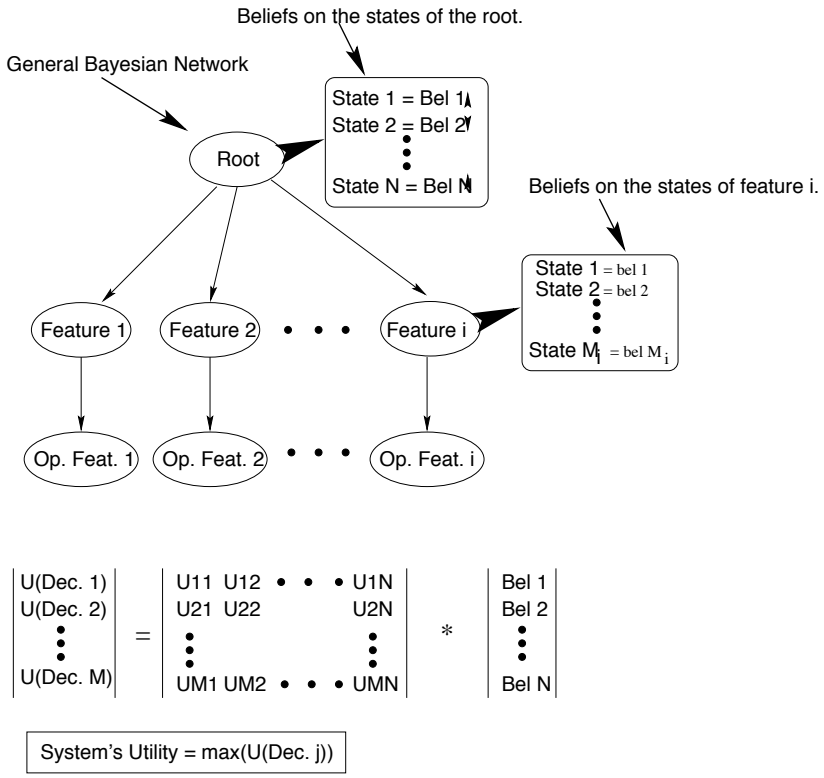

Figure 5.1. A generic Bayesian network for the Ascender II system. The network shows the beliefs for a generic feature and for the root node, as well as the expression used to compute the system's utility.

feature "i" has "M" states, $state_1, state_2, \cdots, state_M$; and each state in feature "i" has a corresponding belief, $bel_1, bel_2, \cdots, bel_M$, corresponding to the current expectation about each state of feature "i". Set the outcome of feature "i" to $state_1$ (i.e. make the belief of $state_1 = 1$ and the belief of all other states equal to 0), and propagate the information through the network. This will change the beliefs in the states of the root node. Use this new set of beliefs in the root node to compute the new utility of the system ($max_i(EU(DR_i|E, F_m)$ in equation 5.4). Multiply this value by bel_1 (the "probability" that feature i is in state 1 ($P(F_m)$ in equation 5.4)). Repeat this process for all states of feature i and then compute the expected utility of the system given feature i ($EU(DR_i|E, F_m)$ in equation 5.4). When completed, the value of information for feature i is found from equation 5.5. The process is repeated for all features and the one with the highest value of information is selected.

5.1.2 The Recognition Process

The recognition process using utility theory has two steps. In the first step the system decides when to stop the reasoning process, and, in the second step, the system decides about the region's label. To stop the reasoning process, the system checks the features that have not been measured yet and computes their corresponding value of information as described above. If none of the features has a positive value of information, meaning that they are not expected to increase the current utility of the system, the reasoning process stops. Otherwise, a feature is selected as described earlier and the process continues. To label a region the system computes the utility of the system after the reasoning process is stopped and, using equation 5.3, labels the region using the decision that has the highest utility. Notice that the final label depends on the utility table which has to reflect the user's preference or the system's goals.

5.1.3 New Networks

The following changes were made in the system for the tests using utility theory. As mentioned in Chapter 3, one problem with the networks used in the first experiments was that the operators were considered completely reliable. This is not true and may cause errors when the value retrieved by an operator is close to a threshold value. Nodes to model the operator's reliability were therefore implemented in the networks. A new network was added in the hierarchy to model Parking Lots at level 1. This network also adds new objects to the system, such as large trucks or RVs. Finally, the structure of the network at level 0 was changed to better represent the geometric features of a region, so that features such as width, area, and the ratio of width and length, were combined in the network and not considered completely independent sources of information as before.

The networks used in these experiments are shown in Figures 5.2 to 5.6. The classes used in an operator's node are the same as the classes in a feature's node, so the classes for the feature's nodes were omitted in the figures.

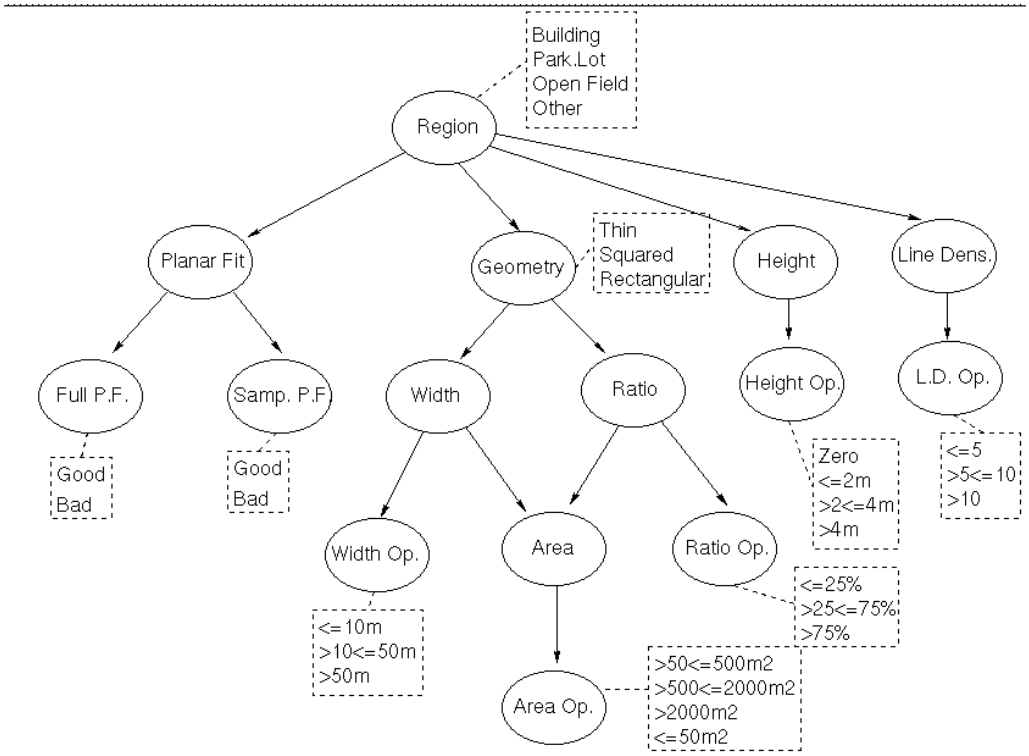

Figure 5.2. The level 0 network determines if a region belongs to one of the possible classes.

5.1.4 How the Ascender II System Works Using Utility Theory - Snapshots

In this section the selection of operators and the decision process using utility theory will be described in detail. As a specific example, consider the boat parking lot in the Avenches data set (see Figure 5.7). The decision table used in the level 0 network (see Table 5.2) was obtained as described earlier (see section 5.1.1). The table reflects the fact that only the correct classification is desired, and that the object classes have the same importance.

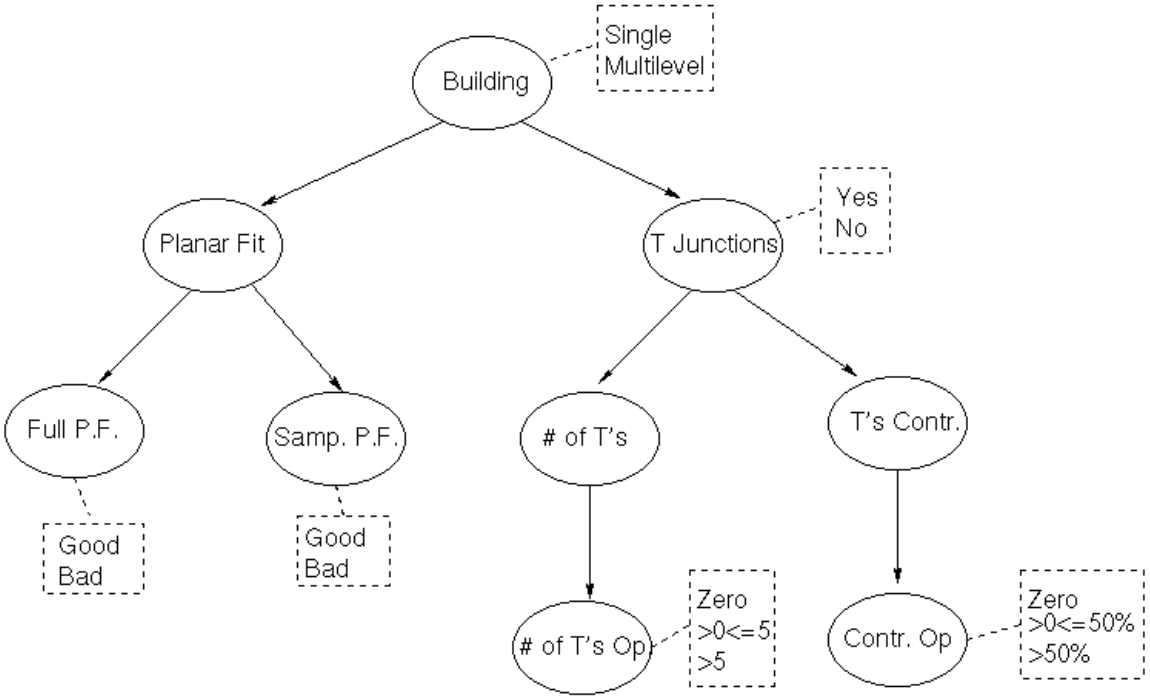

Figure 5.3. The level 1 network is invoked for each building found in level 0 and tries to determine if it is a single-level building or a multilevel building.

Using the utility table and the probabilities given by the beliefs the utility of the system was computed. First, the expected utility of each decision available was calculated using equation 5.1. The utility of each decision is shown in Table 5.3.

The current utility of the system is 0.38, the highest utility available, and the best decision without getting any evidence is to label the parking lot region as a building. The reasoning process is started by computing the value of information of each feature using equations 5.4 and 5.5.

To determine the expected utility for the feature "Planar fit" the current beliefs are: P(Good)=0.5545 and P(Bad)=0.4455. Simulating a "Good" planar fit in the network (setting the feature Planar Fit to Good with probability 1 and propagating this information through the network) leads to a new set of beliefs in the root node (Table 5.4). The process is repeated for a "Bad" planar fit and a second set of beliefs

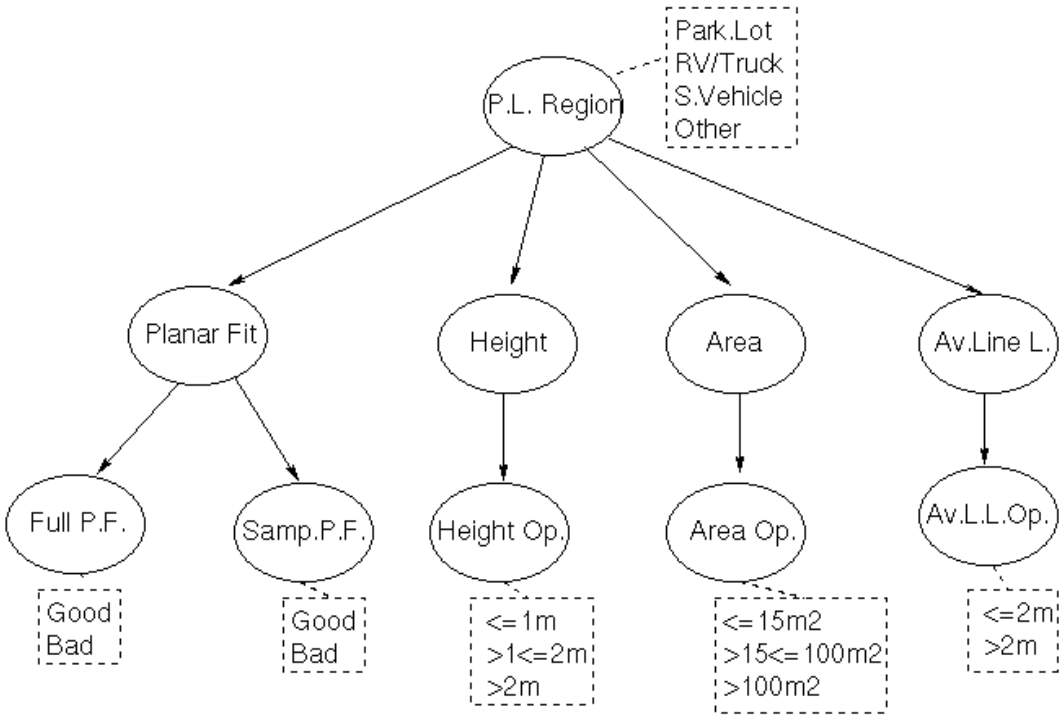

Figure 5.4. The level 1 network is invoked for each parking lot found in level 0 and tries to refine its classification.

is obtained for the root node (Table 5.4). The utility of the system can be computed for each case (Good and Bad); this gives the values 0.3895 for a "Good" planar fit and 0.4209 for a "Bad" planar fit. The expected utility (equation 5.1) of the feature "Planar Fit" is:

$$EU(PlanarFit) = 0.3895 * 0.5545 + 0.4209 * 0.4455 = 0.4035$$

Repeating the process for each feature gives the expected utility values shown in Table 5.5. The feature with maximum expected utility (Height) is selected. The height operator is then applied to the region and returns an average height of 1.1 meters. This value is compared with the ranges in each state of the node *Height Operator* and one state is selected as evidence. The evidence is propagated through the network giving the beliefs shown in Table 5.6 for the classes in the root node.

Table 5.2. The decision table shows all utilities for the level 0 network in the Ascender II system and the initial beliefs for each class in the root node.

Decisions	*Class*			
	Building	Parking Lot	Open Field	Other
Decide Building	1	0	0	0
Decide Parking Lot	0	1	0	0
Decide Open Field	0	0	1	0
Decide Other	0	0	0	1
Probability(Class)	0.38	0.25	0.27	0.10

Table 5.3. The utility value of each decision before acquiring any evidence for the parking lot of boats in the Avenches data set.

Decisions	Utility
Decide Building	0.38
Decide Parking Lot	0.25
Decide Open Field	0.27
Decide Other	0.10

Table 5.4. Planar Fit. The beliefs for each class in the root node when the class "Good" is entered for the feature Planar Fit, and when the class "Bad" is entered for the feature Planar Fit.

Class	Probabilities given Good	Probabilities given Bad
Building	0.3895	0.3232
Parking Lot	0.1127	0.4209
Open Field	0.3895	0.1212
Other	0.1083	0.1347

Table 5.5. The expected utility value of each feature. In this case the feature Height has the highest expected utility and should be selected.

Features	Expected Utility
Planar Fit	0.4035
Height	0.6700
Line Density	0.3960
Width	0.3600
Ratio	0.3956
Area	0.3600

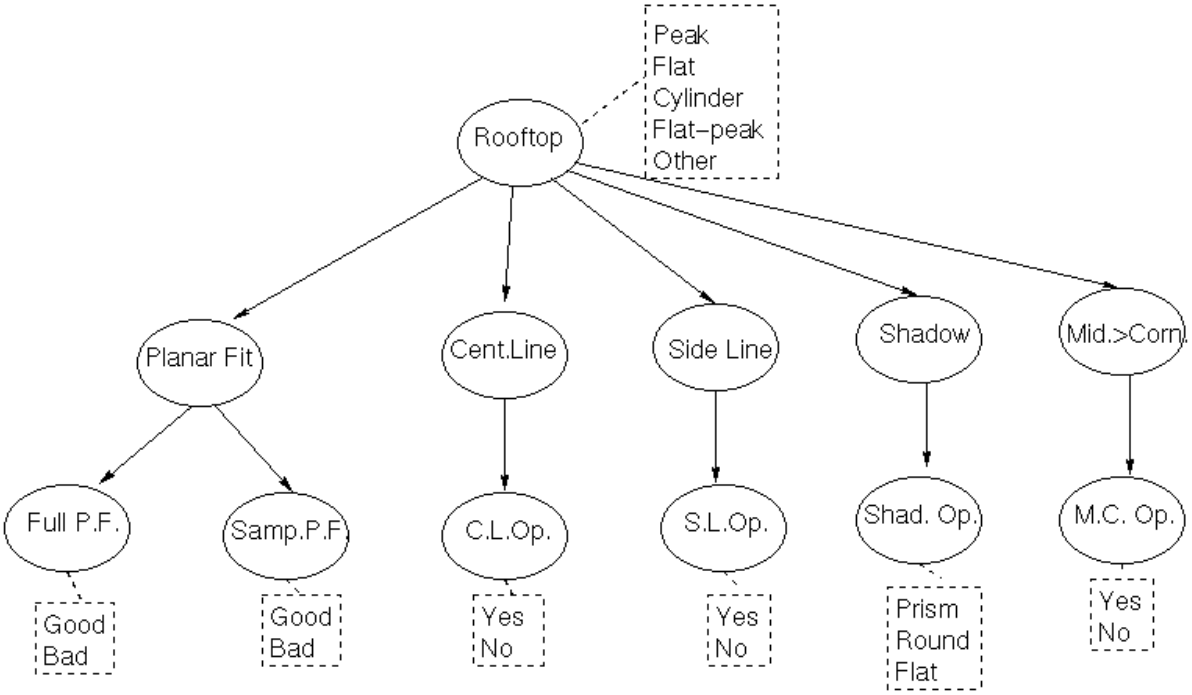

Figure 5.5. This network is called after a single building is detected and it is used to determine the rooftop type.

Table 5.6. The beliefs for each class in the root node after evidence for average height equals to 1.1 meters is entered and propagated through the network.

	Building	Parking Lot	Open Field	Other
Probability(Class)	0.11	0.57	0.22	0.10

Using this new set of beliefs a new utility for the system (value equals 0.57) is computed using equation 5.2, and the process is repeated to select the next operator. The new utilities for each feature not measured yet are shown in Table 5.7. This time the Planar Fit feature was selected. The operator returns a value of 41% for a good planar fit (which means that the fit is not very good). This value is entered as evidence (in terms of likelihood) as follows: multiply the current belief in a good planar fit in the *Planar Fit Operator*'s node by the likelihood of 0.41 (the value returned in terms of probability), and the belief in a bad planar fit by (1 - 0.41), the likelihood of a bad

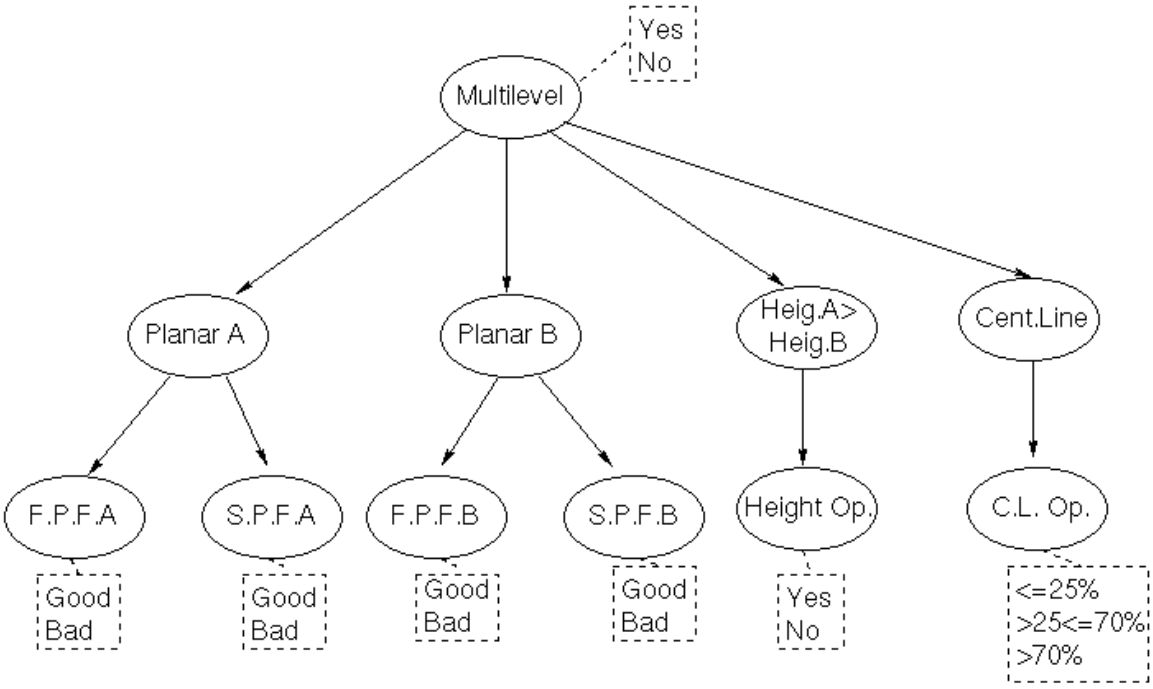

Figure 5.6. This network is called when a multilevel building is detected. The reasoning system calls this network to confirm the recognition at level 1.

planar fit. The new set of beliefs in the *Planar Fit Operator*'s node is propagated through the network, giving a new set of beliefs for the classes in the root node, as shown in Table 5.8

The system's utility moved from 0.57 to 0.60. The new expected utility for each feature is computed again and these values are shown in Table 5.9. As none of the features have an expected utility greater than the current utility, the system stops the reasoning process at level 0. Using the current beliefs and the utility table, the Ascender II system labels the region as a "Parking Lot" and proceeds to level 1. The decision table for the network at level 1, for Parking Lot, is shown in Table 5.10.

The expected utility for each feature at this level is shown in Table 5.11. The feature "Area" is selected and the operator returns a value of 1133 square meters.

Table 5.7. The expected utility value of each feature after the measure of feature Height. In this case the feature Planar Fit has the highest utility and should be selected.

Features	Expected Utility
Planar Fit	0.6035
Line Density	0.5700
Width	0.5700
Ratio	0.5700
Area	0.5700

Table 5.8. The beliefs for each class in the root node after evidence for height and planar fit (good planar fit = 0.41) are entered and propagated through the network.

	Building	Parking Lot	Open Field	Other
Probability(Class)	0.10	0.60	0.20	0.10

Table 5.9. The expected utility value of each remaining feature after the measurements of Height and Planar Fit. In this case none of the features has an expected utility higher than the system's utility and the process stop.

Features	Expected Utility
Line Density	0.600
Width	0.600
Ratio	0.600
Area	0.600

Table 5.10. The decision table shows all utilities in the level 1 network for Parking Lots in the Ascender II system, and the initial beliefs (a-priori probabilities) for each class in the root node.

Decisions	*Class*			
	Parking Lot	Truck-RV	Single Vehicle	Other
Decide Parking Lot	1	0	0	0
Decide Truck-RV	0	1	0	0
Decide Single Vehicle	0	0	1	0
Decide Other	0	0	0	1
Probability(Class)	0.35	0.40	0.15	0.10

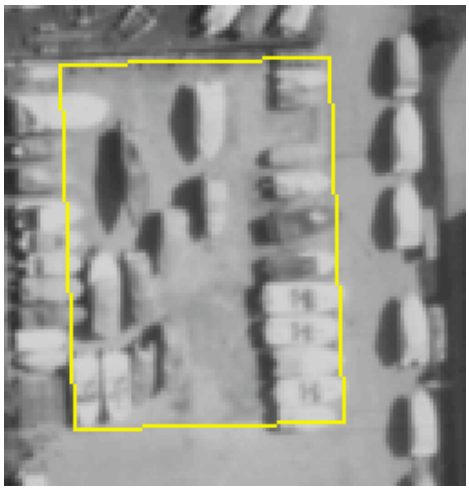

Figure 5.7. The boat parking lot in the Avenches dataset is used to illustrate how utility theory can be used for feature selection and recognition.

Table 5.11. The expected utility value of each feature.

Features	Expected Utility
Planar Fit	0.5800
Average Line Length	0.4500
Area	0.6550
Height	0.5950

This value is entered as evidence and propagated through the network. The new beliefs for the classes in the root node are shown in Table 5.12

The new system's utility is 0.83. Again, the expected utility for each feature is computed (see Table 5.13) and as the maximum expected utility is not greater than the system's current utility, the reasoning process is interrupted at this level. Since there are no more levels of reasoning for the Parking Lot branch, the Ascender II system labels the region as a *Parking Lot*.

Table 5.12. The beliefs for each class in the root node after evidence for the feature "Area" with value 1133 square meters is entered and propagated through the network.

	Parking Lot	Truck-RV	Single Vehicle	Other
Probability(Class)	0.83	0.12	0.04	0.01

Table 5.13. The expected utility value of each feature.

Features	Expected Utility
Planar Fit	0.83
Average Line Length	0.83
Height	0.83

5.2 Experiments Using Utility Theory

The experiments described here were performed over the same data sets used in the experiments in Chapter 4. For performance comparison, the networks shown in section 5.1.3 were also used in the Ascender II system, using uncertainty distance for feature selection and a relative threshold for region recognition. These results are presented at the end of this section and summarized in Tables 5.17, 5.18, and 5.19. Again, the network at level 0 (Figure 5.2) attempts to recognize the region's class. If the class identified is *Building* or *Parking Lot*, the process proceeds to the next level; otherwise it is stopped and a generic 3D object class model is selected from the database for visualization purposes. The reasoning process through the networks is the same as described in Chapter 3; the only difference is that a new branch was added for Parking Lots at level 1. In the case where a parking lot is identified in the network at level 0, the network at level 1 is called (Figure 5.4) and the system attempts to classify the region as a real parking lot, a single vehicle, or a Truck-RV.

5.2.1 Results

The results presented here were obtained over the three data sets presented in Chapter 4, using the same set of input regions. The prior probabilities used in these experiments are shown in Tables 5.14, 5.15, and 5.16. The average number of calls per region, in all three data sets, is shown in Table 5.17. The column "Total" shows the overall average number of calls for the 79 regions; the column "Total2" shows the overall average number of calls only for the regions correctly classified at the level 0 network. The specific columns (Building, Parking Lot and Open Field) also show

Table 5.14. Probability distribution of beliefs in the root node for the level 0 network among the states Building, Parking Lot, Open Field, and Other for all three data sets.

Knowledge	Build.	P.Lot	O.Field	Other
Fort Hood	0.38	0.25	0.27	0.10
Avenches	0.36	0.32	0.20	0.12
Fort Benning	0.50	0.25	0.15	0.10

Table 5.15. Probability distribution of beliefs in the root node of the level 1 network for Parking Lots among the states Parking Lot, Truck-RV, Single Vehicle, and Other for all three data sets.

Knowledge	Parking Lot	Truck-RV	Single Vehicle	Other
Fort Hood	0.40	0.20	0.30	0.10
Avenches	0.35	0.40	0.15	0.10
Fort Benning	0.25	0.25	0.40	0.10

the average number of calls for the regions correctly classified at the level 0 network. The large difference in the average for the column "Open Field" is due to the more efficient way utility theory selects operators and decides about open fields.

The overall performance of the two systems is summarized in table 5.18. The systems' performance, in terms of classification accuracy, were quite similar, and for the data sets used we can not conclude that one performed better than the other. However, in terms of the number of operators used, the system using utility theory performed better (using a smaller number of operators on average) than the system using uncertainty distance (see Table 5.19). This result was expected, and it confirms our expectations that the application of utility theory improves efficiency. Notice also

Table 5.16. Probability distribution of beliefs in the root node of the level 2 network for Rooftops among the states Flat, Peak, Cylinder, Flat-peak and Other for all three data sets.

Knowledge	Flat	Peak	Cylinder	Flat-peak	Other
Fort Hood	0.45	0.15	0.15	0.15	0.10
Avenches	0.30	0.36	0.12	0.12	0.10
Fort Benning	0.30	0.30	0.15	0.15	0.10

Table 5.17. Average number of calls to vision operators. Total column shows values for all classes in the data sets and Total2 column shows the values for regions correctly classified at Level 0 in the data sets. The other columns show the values for specific classes (Building, Parking Lot and Open Field).

Decision process	Total	Build.	P.Lot	O.Field	Total2
Utility Theory	4.9	6.4	5.2	1.1	5.1
Uncertainty Distance	5.8	6.2	6.9	4.6	6.0

Table 5.18. Summary of the recognition process for different data sets. In each case the number of objects correctly identified is shown, followed by the total number of objects evaluated by the system.

Uncertainty Distance				
Data set	Overall	Level 0	Level 1	Level 2
Fort Hood	34/42	36/42	22/24	21/21
Avenches	12/18	15/18	12/13	5/7
Fort Benning	17/19	18/19	17/18	17/18
Total	63/79	69/79	51/55	43/46
Utility Theory				
Data set	Overall	Level 0	Level 1	Level 2
Fort Hood	35/42	37/42	22/24	21/21
Avenches	13/18	16/18	12/13	5/7
Fort Benning	16/19	18/19	17/18	16/17
Total	64/79	71/79	53/55	42/45

that the number of operators used by the system in both cases (uncertainty distance and utility theory) is about half the number of operators available within the system.

5.3 Adding Cost to the Visual Operators

The visual operators have been selected based only on their value of information, as given in equation 5.5. The cost of applying an operator was ignored; all operators

Table 5.19. Total number of calls to visual operators for all data sets for all classes.

Decision process	Number of Operators
Utility Theory	430
Uncertainty Distance	475
All Operators	906

were assumed to have zero cost. This is not generally true and the system should be able to incorporate varying costs into its algorithm selection process. These costs may be based on processing time, algorithm complexity, resources required, etc. In the following paragraphs some cost functions will be presented with a brief discussion whether these functions can be used or not in the current system.

A cost based on processing time requires a mapping function that outputs the running time of an operator given the number of pixels to be processed. In this scenario, ideally, all algorithms should have their implementation optimized. A badly implemented operator, in this case, could be useless and never be selected. The operators in the Ascender II system were defined and implemented considering only the output they provide, thus their implementation are not necessarily optimal in terms of running time. For instance, Figure 5.8 shows the three steps of the line density operator as implemented here. The operator goes over all lines previously extracted in the image, filtering the lines that are inside a region and saving the filtered lines in a file. The algorithm then reads the file and computes the line density inside the region. The process of saving the filtered lines into a file and then immediately reading it is not really necessary. This implementation could be optimized and the processing time could be considerably reduced. Thus, this type of cost function can not be used in the current system.

A cost based on resources required, such as memory, processor speed, or number of processors, would not apply in our case. It is true that each operator requires some memory to compute a feature over a region, but it is bounded by the image size, which is kept in memory all the time. The operators run on the same machine, so they all use the same processor speed. None of them is a parallel operator and they all run on a single processor machine. Thus, this type of cost function is not adequate for the current system.

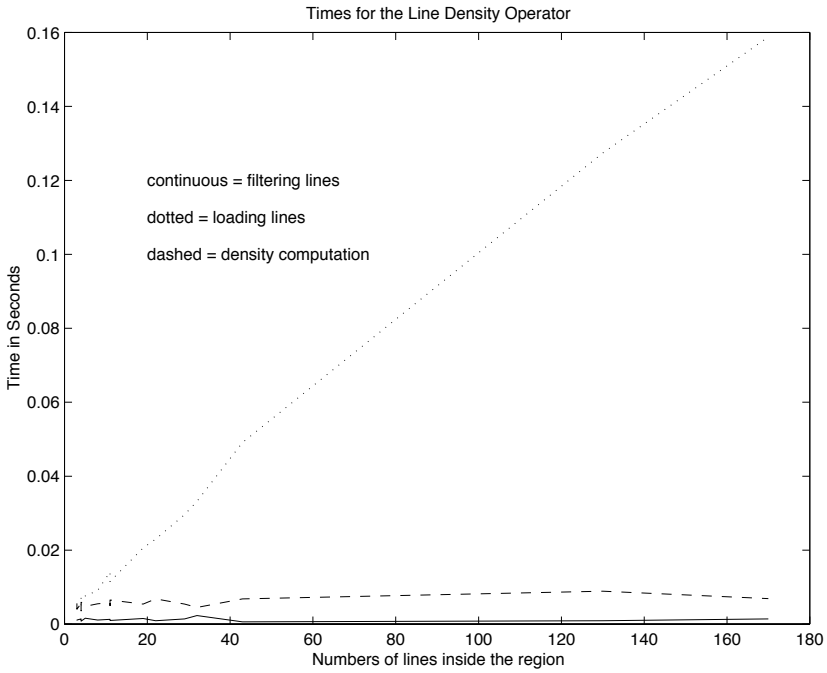

Figure 5.8. The graph shows the time required to compute line density in the Avenches data set. The operator is divided into three parts, filtering the lines that are inside the region and saving them into a file, loading the lines from a file, and computing the density inside the region.

The cost function used in Ascender II is based on the algorithmic complexity of each operator. The operators were divided into three complexity classes and an associated cost was attributed to each of these classes. A short description of each operator is presented in Table 5.20. The first class has all operators which compute their output based only on the information from a small set of pixels. Thus, these operators run in constant time and they belong to the complexity class O(1). The second class has the operators which need to compute features based on the lines found in the image and stored in the visual subsystem. The running time of these operators is linear with the number of lines in the image and they form the complexity class O(L). The third complexity class has the operators which need information from all pixels inside the region being processed to compute a feature. The running time of these operators is linear with the number of pixels being processed. These operators form the O(N) complexity class.

Table 5.20. List of visual operators, their corresponding complexity classes, and a short description of each of them.

Operator	*Class*	*Description*
2	O(L)	Computes # of T junctions inside the terminal areas (see Figure 4.13) in the region.
3	O(L)	Computes contrast of T junctions in the terminal areas (see Figure 4.13) of the region.
6	O(N)	Computes a Planar Fit over the DEM of the region and returns the correlation of this fit.
8	O(L)	Computes the Line Density inside the region given as number of lines per 100 m^2.
10	O(N)	Computes the Average Height of the region in meters.
11	O(1)	Computes the Ratio in % between width and length.
13	O(1)	Computes the Width of the region in meters.
15	O(L)	Finds if there is a line in the center of the region and returns the % of this line compared to the length of the region.
17	O(L)	Same as 15 but returns if the % is larger than 65% or not.
18	O(N)	Computes the shape of the shadow of the region.
19	O(1)	Computes the difference in height between the center of the region and the corners of the region.
20	O(N)	Computes the correlation of the DEM inside the region against a Peak roof model [34].
21	O(N)	Same as 20 for a Flat-peak model
22	O(N)	Same as 20 for a Flat model
23	O(N)	Same as 20 for a Cylinder model
24	O(L)	Checks if there exists a line in the sides of the region.
25	O(1)	Computes the area of the region

5.3.1 Defining a cost associated with a complexity class

The cost associated with the complexity class of an operator could be expressed in terms of money, time, number of pixels processed, or any other relevant scale. The problem is that no matter what scale is used the cost has to be expressed in terms of utility so it can be combined with the "information-based" utility used up until this point.

Without cost the value of information (equation 5.5) considers only the increment in the system's utility given by the knowledge of a certain feature. In this case the value of information can be defined as the expected gain, in terms of utility, that we can get if we know the value of the feature. Then, equation 5.5 could be written as:

$$Gain(F_m) = EU(DR_{\alpha'}|E, F_m) - EU(DR_{\alpha}|E) \tag{5.6}$$

If α' is the same as α and $EU(DR_{\alpha}|E) < 1$, than the $Gain(F_m)$ is always positive. It will be zero when $EU(DR_{\alpha}|E) = 1$. Notice that because we select the operator with the highest value of information (or expected gain), the function of the expected gain with the order in which an operator is applied is a positive, decreasing function, as represented in Figure 5.9. Thus the later in the reasoning sequence that the operator is applied the smaller the increase in the system's utility (or gain) given by the operator.

When using cost, the value of information is computed using one of the equations below:

$$VI(F_m) = Gain(F_m) - Cost(F_m) \tag{5.7}$$

or

$$VI(F_m) = EU(DR_{\alpha'}|E, F_m) - (EU(DR_{\alpha}|E) + Cost(F_m)) \tag{5.8}$$

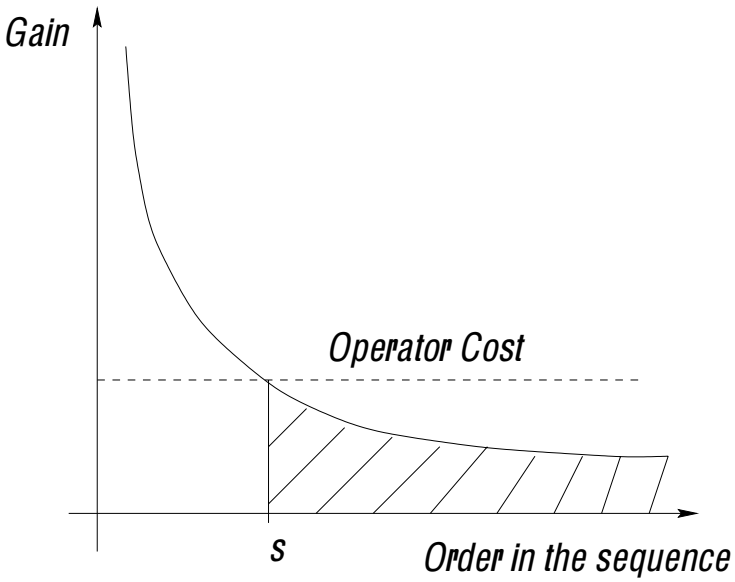

Figure 5.9. The graph shows how the gain in the utility of the system decreases as a function of when an operator is applied. The dashed line represents the cost of the operator and beyond point s in the sequence the operator is not invoked.

If we look at equation 5.7 we can see that the cost for an operator should not be too high or else the operator could have negative value of information and would never be called, making it useless and forcing the system to use only cheap and perhaps unreliable operators, leading to poor classifications. On the other hand, the cost of an operator should not be too low, or else the system would behave as if all operators were zero cost operators, leading to a system that could select very expensive operators, even when they would not be required.

The operator's cost was defined in terms of utility values based on the operator's complexity class. We define the maximum cost (MC) of an operator as a linear function of the mean of the expected gain given by an operator over the sequence of operators (called average gain or AG), that is, $MC = F * AG$.

The cost function for each complexity class was defined as a linear function of the maximum cost, as presented in Table 5.21. The average gain was computed over all regions in the Avenches data set for all cases where evidence over the sequence of operators were converging, that is, $\alpha' = \alpha$ in equation 5.8. The value of the average

Table 5.21. This table shows the generic function of an operator's cost based on the operator's complexity class.

Complexity	*Cost*
O(1)	A*MC
O(L)	B*MC
O(N)	C*MC

gain obtained is 1.083 out of 10, which was approximated to 1.1. This value was used as a first attempt to define the cost functions in terms of utility. Further experiments were performed to tune the values of F, A, B, and C.

5.3.2 Tuning the cost function

A set of experiments were performed in the Avenches dataset to tune the cost functions in the Ascender II system and to study the system behavior when using cost.

In the first experiment the goal is to determine a good value for F in the expression $MC = F * AG$. A first guess was made for the constants A, B and C in the cost functions presented in table 5.21. The value of the constants were defined as $A = 0.02$, $B = 0.4$, $C = 1.0$. The value of AG was set to 1.1 and the value of F was changed from 0 to 1; for each value of F the operators selected by the system were applied in the Avenches data set and the regions were classified properly. The resulting curve is shown in Figure 5.10. The number of operators applied in the image was reduced when the cost factor (F) increased, as expected. The average of the final beliefs in each level did not change significantly (less than 3%), and the final classification of the regions for the Avenches data set was the same. The use of cost proved to be very efficient, the system kept the same accuracy and used less resources to perform the same task. Figure 5.10 shows a clear turning point when $F = 0.5$, at this point the number of operators used still decreases but the rate at which it decreases is smaller

than before (less than 5%). The value of F could be selected as any value between 0.5 and 1.0 without any significant change, thus we set the value of F as 0.5.

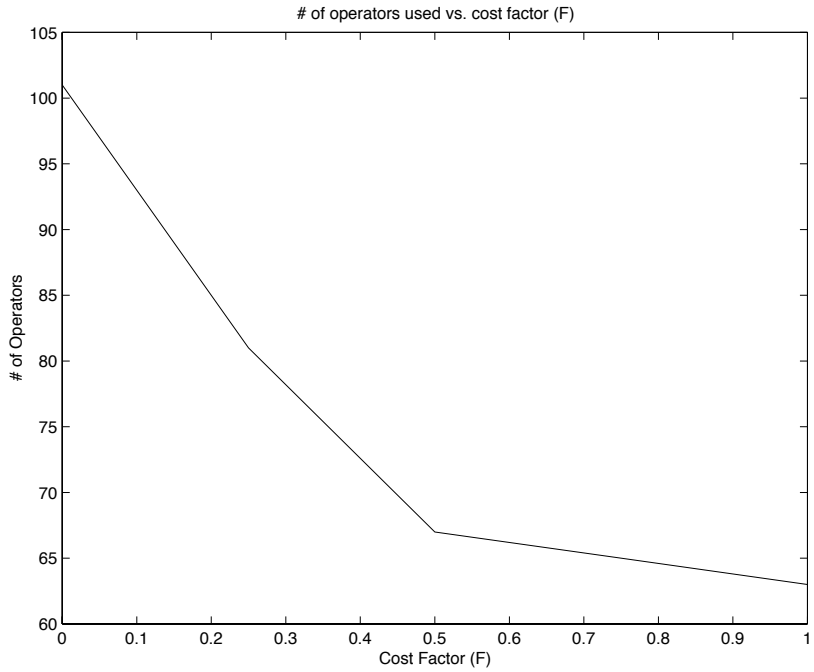

Figure 5.10. The graph shows the number of operators used by the system in the Avenches data set when the value of F goes from 0 to 1. For the Avenches data set the number of regions correctly classified was the same.

In the second experiment the constant value B in the cost function presented in Table 5.21 for the complexity class $O(L)$ was investigated. For any region we know that $O(1) < O(N)$. We can also say that for all lines inside a region $O(1) < O(L) < O(N)$. The problem, in our case, is that the $O(L)$ operators consider all lines in the image and, as a first step, clips the lines in the image against the region boundaries. Because of this step it is hard to compare an $O(L)$ operator with an $O(N)$ operator. If the region is small and the image has a large number of lines we can find that $O(L) > O(N)$. In this experiment we used F as 0.5 (as defined in the previous experiment), we kept AG, A, and C with the same values used in the first experiment and we changed the value of B from 0.02 (the constant value in the cost

function for the $O(1)$ complexity class) to 1.5 (50% more expensive than the cost for an $O(N)$ operator).

The result is shown in Figure 5.11. The average final beliefs in each level have not changed significantly (less than 1%) and the final classification of the regions were the same. When the cost factor of an $O(L)$ operator is between 0.2 and 1, the variation in the total number of operators used by the system is small (67 to 66). Considering that the overall variation in the total number of operators used is not too large (around 15%) and considering the time required to the $O(L)$ operators was, on average, smaller than the time required to the $O(N)$ operators in the Avenches data set we decided, empirically, to set the value of the constant B in the cost function of the $O(L)$ operators to 0.4 (our guess in the first experiment, thus validating our choice of B).

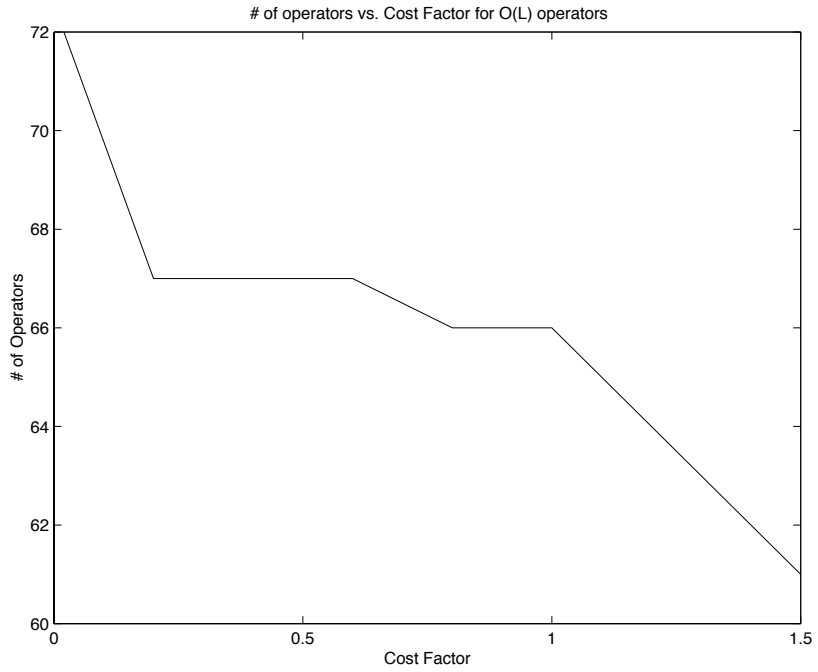

Figure 5.11. The graph shows the number of operators used by the system in the Avenches data set when the constant B in the cost function for the O(L) operators goes from 0.02 to 1.5. The overall classification did not change.

Table 5.22. This table shows the function of an operator's cost based on the operator's complexity class.

Complexity	*Cost*
O(1)	0.02*MC
O(L)	0.4*MC
O(N)	1.0*MC

Based on these two experiments and in the fact that an $O(1)$ operator should cost much less than an $O(N)$ operator we also decided, empirically, to fix the values of the constants A and C in the cost function for the complexity classes $O(1)$ and $O(N)$ as 0.02 and 1 respectively.

In order to define the cost functions for each complexity class in the Ascender II system only the average gain (AG) value remains to be investigated. In this third experiment we used $F = 0.5$ and the cost functions as defined in Table 5.22; the value of AG was changed from 1.1 to 10 and the results are presented in Figures 5.12, 5.13, and 5.14.

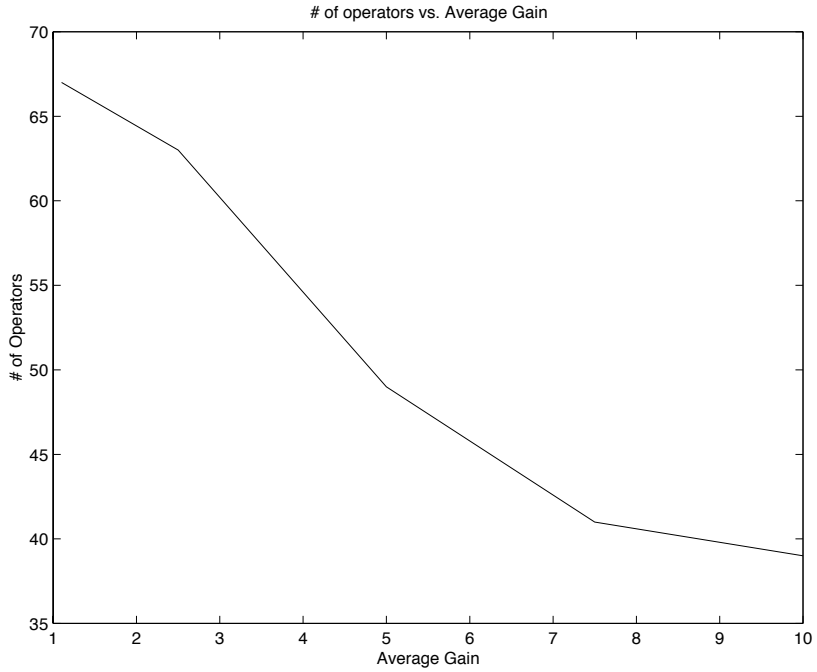

Figure 5.12. The graph shows the number of operators used by the system in the Avenches data set when the average gain goes from 1.1 to 10.

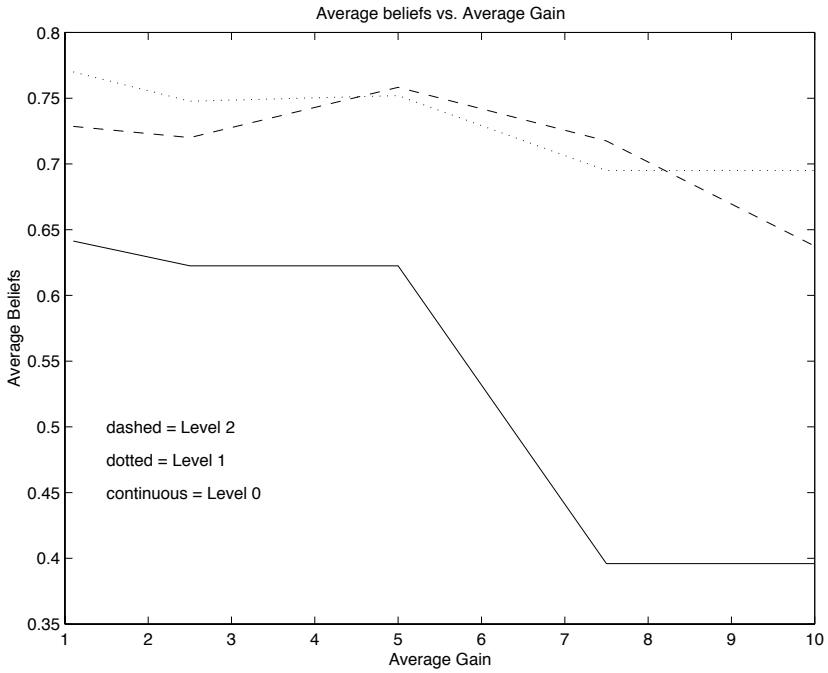

Figure 5.13. The graph shows the average beliefs for each level in the Avenches data set when the average gain goes from 1.1 to 10.

Figure 5.12 shows that as the average gain increases, the number of operators used decreases, but when the average gain increases above 5 the number of regions correctly classified decreases (Figure 5.14). In particular, there was a dramatic decrease in the beliefs at level 0 when AG increased beyond 5. This happened because the system exchanged an expensive operator (Height) by another one, not so expensive but also less reliable (Line Density). Because the system started to use cheaper operators the evidence gathered about the object classes was not so strong and the average beliefs in each level of inference decreased (Figure 5.13).

5.3.3 Running the Ascender II system using cost

From the experiments just described the maximum cost of an operator was defined as $MC = 0.5 * AG$. The cost function for each complexity class was kept as presented in Table 5.22. From the experiments performed in the Avenches dataset the value of the average gain can be any number between 1.1 and 5, because the average final

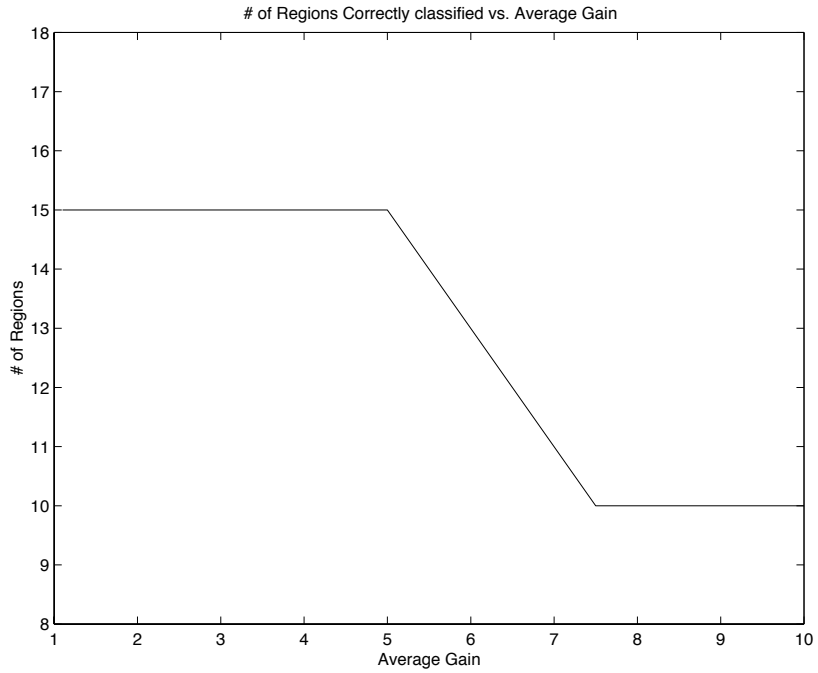

Figure 5.14. The graph shows the number of regions correctly classified in the Avenches data set when the average gain goes from 1.1 to 10.

beliefs in each level are about the same and the classification of the regions was the same. However, the higher the value of AG in this range did dramatically lower the number of operators used. So an AG value of 5 would be the best for the Avenches data set.

Experiments were performed using the Ascender II system with operator's cost in the other two datasets (Fort Benning and Fort Hood). Although we should have used a value of 5 for AG, this value proved to be too high for the Fort Benning data set, decreasing the number of regions correctly classified considerably (values not presented here). Two sets of experiments were performed; in the first experiment we used an average gain value of 1.1, and in the second experiment we used an average gain value of 2.5. The cost for each complexity class in this case is shown in Table 5.23.

The results are presented in Tables 5.24, and 5.25. The introduction of cost in the Ascender II system has a small change in the final classification of the regions

Table 5.23. Cost for each complexity class and for each average gain used.

Complexity	*Average Gain = 1.1*	*Average Gain = 2.5*
O(1)	0.011	0.025
O(L)	0.220	0.500
O(N)	0.550	1.250

Table 5.24. Number of operators used in each data set when the average gain changes.

Data Set	*Average Gain = 0*	*Average Gain = 1.1*	*Average Gain = 2.5*
Fort Hood	178	119	114
Avenches	101	67	63
Fort Benning	113	100	97

(1.3 to 2.6%), but it shows a significant decrease in the number of operators used (27 to 30%). The changes in the average final beliefs in each level was too small to be significant (less than 3%), and not considered.

5.4 Changing Preferences in the Reasoning Subsystem

One of the problems that was mentioned at the beginning of this chapter was the fact that personal preferences could not be easily implemented when using uncertainty distance. In this section we will show how personal preferences can be adjusted when using utility theory.

Table 5.25. Number of regions correctly classified in each data set when the average gain changes.

Data Set	*Average Gain = 0*	*Average Gain = 1.1*	*Average Gain = 2.5*
Fort Hood	36	36	35
Avenches	15	15	15
Fort Benning	15	14	13

5.4.1 Setting preferences - background

The key idea on setting preferences in a utility table is to understand how decisions are made when we use utility theory. The decision is made just by maximization, that is, selecting the decision that has a maximum value in a decision vector (equation 5.2 and 5.3). The decision vector is obtained by multiplying the matrix, composed of utilities, by the column vector, composed of the beliefs in each object class, as shown in equation 5.9.

$$\begin{vmatrix} Dec_1 \\ Dec_2 \\ \vdots \\ Dec_n \end{vmatrix} = \begin{vmatrix} U_{11} & U_{12} & \cdots & U_{1m} \\ U_{21} & U_{22} & \cdots & U_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ U_{n1} & U_{n2} & \cdots & U_{nm} \end{vmatrix} * \begin{vmatrix} Bel_1 \\ Bel_2 \\ \vdots \\ Bel_m \end{vmatrix} \tag{5.9}$$

Setting personal preferences means adjusting the values in the utility table such that, for a given set of values in the belief vector, if the person prefers $Decision_i$ for this distribution of probabilities over the set of events, then $Decision_i$ will be selected. Notice that each column in the utility table is directly related to an event, the belief vector gives the probabilities of any possible event to be true, and the decision vector gives the value of each decision.

Suppose, for instance, that we have the following situation: there are two decisions that we are interested (say Dec_p, and Dec_q). Suppose there are 3 events that are related to these decisions (say $event_i$, $event_j$, and $event_k$). If we look to equation 5.9, we are interested only in the part presented in equation 5.10.

$$\begin{vmatrix} Dec_p \\ Dec_q \end{vmatrix} = \begin{vmatrix} U_{p1} & U_{p2} & \cdots & U_{pi} & U_{pj} & U_{pk} & \cdots & U_{pm} \\ U_{q1} & U_{q2} & \cdots & U_{qi} & U_{qj} & U_{qk} & \cdots & U_{qm} \end{vmatrix} * \begin{vmatrix} Bel_1 \\ Bel_2 \\ \vdots \\ Bel_i \\ Bel_j \\ Bel_k \\ \vdots \\ Bel_m \end{vmatrix} \tag{5.10}$$

In this case, because we are interested only in these two decisions and they are related only to events i, j, and k, we can set all the utilities in these two rows to zero, except for U_{pi}, U_{pj}, U_{pk}, U_{qi}, U_{qj}, and U_{qk}. Then and we have:

$$Dec_p = U_{pi} * Bel_i + U_{pj} * Bel_j + U_{pk} * Bel_k$$

$$Dec_q = U_{qi} * Bel_i + U_{qj} * Bel_j + U_{qk} * Bel_k$$

Lets now make an analysis of the decisions based on each event: if we want to make decision p when event i is true, we have to make $U_{pi} > U_{qi}$. If we want to make decision q when the event j is true, we have to make $U_{qj} > U_{pj}$. Finally, if we want either decision p or decision q when event k is true, then we have to make $U_{pk} = U_{qk}$. Reasoning this way we can adjust all set of utilities in the utility table according to our personal preferences.

In the Ascender II system we have a particular condition where a decision is selected based only on one event, and also, there is only one event for each decision. Thus, the utility table is a square matrix ($n = m$).

$$\begin{vmatrix} Dec_1 \\ Dec_2 \\ \vdots \\ Dec_n \end{vmatrix} = \begin{vmatrix} U_{11} & U_{12} & \cdots & U_{1n} \\ U_{21} & U_{22} & \cdots & U_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ U_{n1} & U_{n2} & \cdots & U_{nn} \end{vmatrix} * \begin{vmatrix} Bel_1 \\ Bel_2 \\ \vdots \\ Bel_n \end{vmatrix} \tag{5.11}$$

If, as we have been doing, we defined the same value, different than zero, for the diagonal entries in the utility matrix, and zeros for all other entries, the decision will be made based only on the belief values. The highest belief among the n values will give the decision. The decision is made by maximizing:

$$max(U_{11} * Belief_1, U_{22} * Belief_2, \cdots, U_{nn} * Belief_n)$$

but as $U_{11} = U_{22} = \cdots = U_{nn}$ the expression above is just:

$$max(Belief_1, Belief_2, \cdots, Belief_n)$$

If the values of the utilities on the diagonal are not the same (but all other entries are still 0), this means that we have introduced a bias towards a decision. Let us assume, without loss of generality, that one of the utility entries in the diagonal is the largest, say U_{22}. Now, even if the belief related to that utility ($Belief_2$) is not the largest in the belief vector, the decision can still be made to choose $Decision_2$. Again the decision is made by maximization:

$$max(U_{11} * Belief_1, U_{22} * Belief_2, \cdots, U_{nn} * Belief_n)$$

However now $U_{22} > U_{11}, U_{33}, U_{44}, \cdots, U_{nn}$. Assuming that $Belief_1$ is the largest in the belief vector, it is still possible that $U_{22} * Belief_2 > U_{11} * Belief_1$, which would lead to $Decision_2$ instead of $Decision_1$.

5.4.2 Setting preferences - general case

If the events were Boolean variables (true or false) and mutually exclusive, we would need to define only one value different than zero for each row in the utility table. In general this is not the case. What we have is a set of beliefs about the events and we have to make a decision based on the combination of different events being possibly true. In this case we have to consider a more general case where the utility table has distinct values in each row.

$$\begin{array}{ccccc} U_{11} & U_{12} & U_{13} & \cdots & U_{1m} \\ U_{21} & U_{22} & U_{23} & \cdots & U_{2m} \\ U_{31} & U_{32} & U_{33} & \cdots & U_{3m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ U_{n1} & U_{n2} & U_{n3} & \cdots & U_{nm} \end{array}$$

Let us consider, for instance, column i, and let us assume, without loss of generality, that U_{2i} is the largest value in column i. This means that if the event corresponding to column i is true, $Decision_2$ will be selected. Consider now the values in a row. Let us, at first assume that:

$$\forall j \sum_{i=1}^{m} U_{ji} = K$$

i.e., the sum of the utility values in each row is a constant. This means that all the decisions have the same weight (no bias is introduced towards any particular decision), and the decision will be made depending on the probabilities of the events. If the utilities in each row add up to different values we are giving a bias to decisions corresponding to higher sums.

Consider the values of the utilities in a particular row, say row i. Suppose that U_{i2} is the largest value in this row followed by U_{i1}. This means that if the event 2 is true

it will make the highest impact towards $Decision_i$. Notice that it might not select decision i because it will depend also on the other values of the utilities in column 2. If instead of event 2 we have event 1 as true, this event will also make an impact towards $Decision_i$ but not as strong as event 2. Combining all the possibilities among utilities and probabilities of events is not simple and, in general, it is an optimization problem which is beyond the scope of this thesis.

As a simple example consider the utility table presented below:

$$\begin{array}{cccc} U_{11} & 0 & 0 & 0 \\ 0 & U_{11}/3 & U_{11}/3 & U_{11}/3 \\ 0 & U_{11}/3 & U_{11}/3 & U_{11}/3 \\ 0 & U_{11}/3 & U_{11}/3 & U_{11}/3 \end{array}$$

We have 4 possible events (4 columns) and 4 possible decisions (4 rows). Let us assume we are mainly interested in event 1, which will lead towards decision 1. If event 1 is not true we don't care about which one is true and what decision will be taken.

If we solve the equation below, we can compute a value for $bel1$.

$$bel1.U_{11} = \frac{U_{11}}{3}(bel2 + bel3 + bel4)$$

$$bel1 = \frac{1}{3}(1 - bel1)$$

$$bel1 = 0.25$$

Thus, in this particular case, $Decision_1$ will be selected if $bel1 > 0.25$ otherwise we could take any other decision randomly.

Any variation in the utility values can be analyzed as we just did here. Now lets investigate what happens when we use some of these utility tables in the Ascender II system.

Table 5.26. The modified utility table shows new values for utilities in the level 0 network.

Decide	*Class*			
	Building	Park. Lot	Open Field	Other
Building	1	0	0	0
Parking Lot	0	0.33	0.33	0.33
Open Field	0	0.33	0.33	0.33
Other	0	0.33	0.33	0.33

Table 5.27. Comparison in terms of number of regions correctly classified per object class and number of operators used per object class (in parentheses).

Utility Table	Building	Park. Lot	Open Field
Original Utility Table 5.1	46 (55)	9 (34)	16 (21)
Modified Utility Table 5.26	47 (50)	9 (32)	16 (22)

5.4.3 Experiments changing preferences

In the first experiment we set the utility table for the level 0 network as shown in Table 5.26; compare this with table 5.1. This new utility table shows that if the final belief in buildings is above 0.25 we want to decide that the region is a building. The results obtained when the system used the new utility table over the three basic data sets (Avenches, Fort Benning and Fort Hood) is presented in Table 5.27.

The system using the original utility table missed three buildings over all data sets (three regions in the Avenches data set which have an average height smaller than 4 m). The system using the modified utility table correctly classified one of these buildings in the Avenches data set while using a smaller number of operators to classify the total set of buildings over the three data sets. Furthermore, no significant difference was noticed for the other object classes.

5.4.3.1 Making strong preferences

In the next set of experiments we used only the Avenches data set. In this case we set the utility table at the level 0 network to classify Parking Lots correctly and

Table 5.28. The table shows the new utilities for the level 0 network, with a preference for Parking Lot areas. Moderate 2-fold preference.

Decide	*Class*			
	Building	Park. Lot	Open Field	Other
Building	5	0	0	0
Parking Lot	0	10	0	0
Open Field	0	0	5	0
Other	0	0	0	5

Table 5.29. The table shows the new utilities for the level 0 network, with a strong preference for Parking Lot areas. Strong 10-fold preference.

Decide	*Class*			
	Building	Park. Lot	Open Field	Other
Building	1	0	0	0
Parking Lot	0	10	0	0
Open Field	0	0	1	0
Other	0	0	0	1

we reduced the utility of all other object classes. The new utility tables are presented in Tables 5.28 and 5.29, specifying a moderate 2-fold and a strong 10-fold preference.

The results obtained when the system used the new utility table with preferences for Parking Lots is presented in Table 5.30.

In this case we noticed two things: the system needed more exploratory calls for operators in order to decide about regions that were not parking lots and, in extreme cases, some of these regions were still misclassified (due to the parking lot decision overriding the correct class). The system also classified the parking lots very fast (using only 1 operator) and misclassified all open fields as parking lots (which was expected). So, by adjusting utilities it is possible to change the system behavior, not

Table 5.30. Comparison in terms of number of regions correctly classified per object class and number of operators used per object class (in parentheses).

Utility Table	Building	Park. Lot	Open Field
Original Utility Table 5.1	7 (15)	6 (19)	3 (5)
Moderate 2-fold Preference for Park. Lots	6 (14)	6 (6)	2 (9)
Strong 10-fold Preference for Park. Lots	4 (23)	6 (6)	0 (3)

only in terms of giving a bias to an object class, but also in terms of the effort required to classify this object class. Notice that when preference for a certain decision is set at a sufficiently high level, the system will not miss any object of that type. All other objects may or may not be classified correctly. Another point in this case is that we observed that objects with the type related to the preferred decision are processed faster than others, using at most 2 operators.

CHAPTER 6

LEARNING MODELS FOR THE CONTROL STRUCTURE

6.1 Learning the structures of the knowledge base

The knowledge engineering necessary to design an efficient Bayesian network (structure and probability tables) is a time consuming task, even for small networks such as those currently used in the Ascender II system. This has been one of the main criticisms of Bayesian networks [12].

Different algorithms for learning Bayesian networks have been developed [13, 23, 18]. Heckerman [27] presents a tutorial on learning and Bayesian networks using different methods to learn the probabilities and network structure. Cooper's algorithm [18] uses unrestricted multinomial distributions to learn probabilities and structures. Friedman's algorithm [23] works with the conditional probability tables and learns local structures. Cheng's algorithms [13] learn only the probability tables or the probability tables and the network structure from data using an information theoretic dependency analysis. We selected Cheng's algorithm because he made available a tool called Power Constructor [14] which implements his algorithms. It is easy to use and it is freely available on the internet.

The first of Cheng's algorithm assumes that the node ordering (dependencies among variables, thus the structure) is given, and has $O(N^2)$ complexity (where N is the number of variables, or nodes, in the network). The second, more general, algorithm requires $O(N^4)$ conditional independence (CI) tests, but node ordering is not required. The general algorithm is similar to the first in the learning phase, but

it has to deal with two major problems: determining if two nodes are conditionally dependent (if so, a link has to be placed between these two nodes), and determining the orientation of the links in the learned structure (arrow direction). The learning occurs in three phases:

- Drafting, where the algorithm computes the mutual information (equation 6.1) of each pair of variables (nodes) in the structure as a measure of closeness in order to determine if two variables are correlated. An edge is added between the two nodes if the value of mutual information is greater than a small value ε. This creates a first draft of the Bayesian network.

- Thickening, where the algorithm adds links to the current structure depending on the results of a group of CI tests based on equation 6.2, where the conditional set C is changed to find conditional dependencies.

- Thinning, where each link in the structure is checked using a group of CI tests (based also on equation 6.2). Some links might be removed from the network if found not to be conditionally dependent.

The mutual information of two nodes X_i and X_j is defined as:

$$I(X_i, X_j) = \sum_{x_i, x_j} P(x_i, x_j) log \frac{P(x_i, x_j)}{P(x_i), P(x_j)} \tag{6.1}$$

and the conditional mutual information is defined as:

$$I(X_i, X_j|C) = \sum_{x_i, x_j, c} P(x_i, x_j, c) log \frac{P(x_i, x_j|c)}{P(x_i|c), P(x_j|c)} \tag{6.2}$$

where C is a set of nodes. When $I(X_i, X_j)$ is smaller than a user defined threshold (ε), we say that X_i and X_j are marginally independent. When $I(X_i, X_j|C)$ is smaller than the threshold (ε), we say that X_i and X_j are conditionally independent given C.

The first expression is used in the Drafting step, and the second expression is used for the group of CI tests in the *Thickening* and *Thinning* steps. More details about the algorithms and the tool can be found in [14].

6.2 Getting data to learn the Bayesian networks

Cheng's algorithm was used to learn the network structure and the conditional probability tables for the variables representing the features and the region label in the hand-crafted networks used in Chapter 5. The operator nodes as well as the operator reliability tables were not included in the learning process, but were added manually when the learning was completed. If the true value of each feature was provided, the tables representing the operator's reliability could also be learned from the data.

The data used for learning was collected from the three data sets described earlier in our research: Ft. Hood, Ft. Benning and Avenches; the 79 input regions presented in Chapter 4 for these datasets were used. These regions represent a mix of objects drawn from buildings, parking lots, grassy fields, etc. All regions were presented to a set of 6 human subjects, and the subjects were asked to estimate the state of each feature in the feature set (features were coarsely quantized to facilitate the human task and to reduce the number of examples required in the learning phase). Consider the building region presented in Figure 6.1. For a region like this the human subject had to answer a set of simple questions, such as:

- Is there a center line in the region? If so, how long is it compared to the length of the region? (Given in percentage).
- What is the height of the region: 0 m? between 0 and 2 m? between 2 and 4 m? more than 4 m?

- What is the width of the region: less than 10 m? between 10 and 50 m? greater than 50 m?

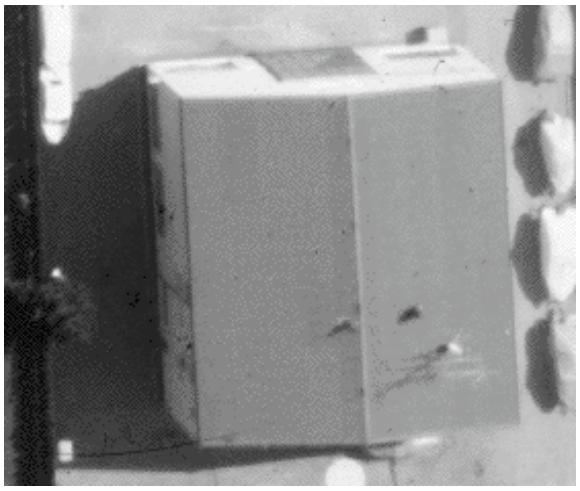

Figure 6.1. A building region in the Avenches data set.

The information obtained from human subjects was compiled and used to learn a Bayesian network at different levels. The learned networks, with the addition of the operator's nodes, are shown in Figures 6.2, 6.3, 6.4, and 6.5. The general structures of the networks are different from the hand-crafted networks, although some of the substructures were preserved (for instance the substructure relating the variables Area, Width, Ratio, Geometry). In the hand-crafted networks features such as planar fit, height and line density, were set as independent (we did not know how to correlate them). In a more detailed analysis we could consider them as dependent, for instance, a high value for line density might increase the expectation of a poor planar fit, and so on. These correlations among features were captured in the learning phase, thus, the learned networks are generally more densely connected.

The networks learned from data are limited to the objects present in the training data. For instance, the data used to learn the networks had only peak- and flat-roofed buildings. Thus the feature *Rooftop* in Figure 6.5 has only states for *Peak* and *Flat*

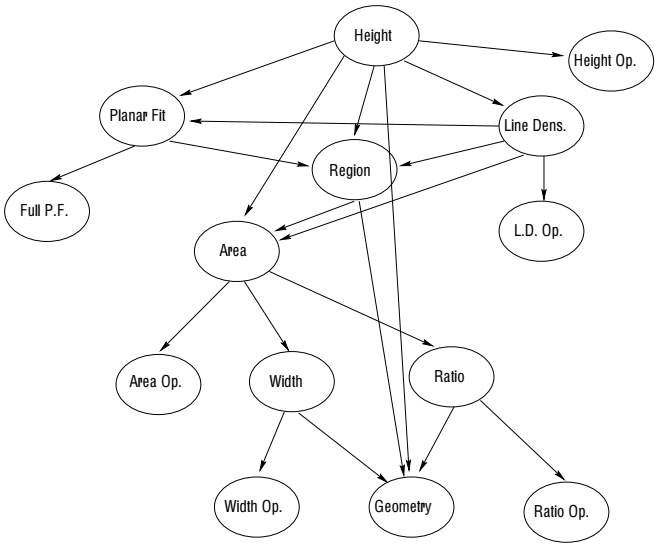

Figure 6.2. The level 0 network learned from data determines if a region belongs to one of the possible object classes: Building, Parking Lot, Open Field, or Other.

roofs, and did not reflect the more general discrimination task as in the hand-crafted networks presented in Figure 5.5.

6.3 Experiments and Results

A set of experiments was designed to demonstrate the performance of the system using the learned networks on the same data sets used for training, as well as on two datasets not used before (Flat Scene and Glandorf datasets). The use of the training data during testing is legitimate because there are two major differences in the procedure that have to be considered:

1. During the experimental phase the features were computed algorithmically from the image data by a visual operator. The results of these computations do not necessarily correspond to the outcome given by humans in the learning phase, for instance, where humans had to classify if the width of a region was less than 10 meters or greater than 10 meters, for values close to 10 meters some humans classified the region width as less than 10 meters while others classified

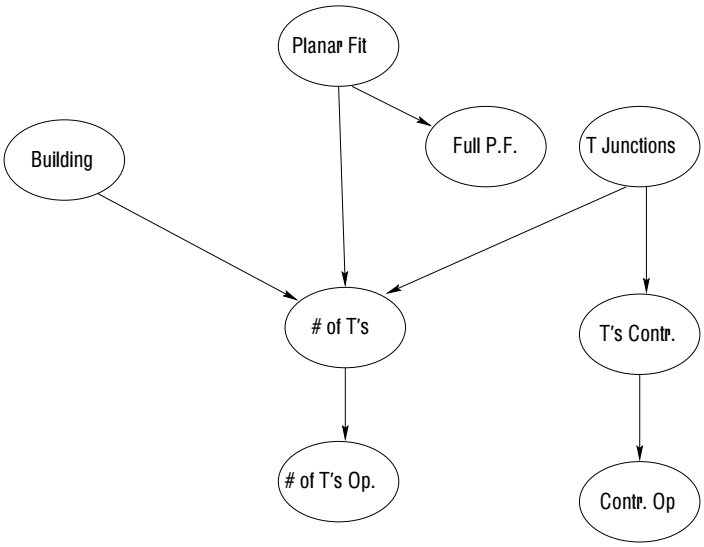

Figure 6.3. The level 1 network for buildings learned from data determines if a region classified as a building is a single level or a multilevel building.

as greater than 10 meters. The visual operator gave a measurement of the region's width.

2. The networks used in the experiments were augmented by the addition of the operator nodes and their corresponding reliability tables. The values of the features computed by the visual operator were entered into the operator's node but were attenuated by the operator's reliability during the propagation.

First, the networks and probability tables (including prior probabilities) as learned from the data (System A) were applied to the 3 datasets (Ft. Hood, Avenches and Ft. Benning). Because the prior probabilities learned from data reflect the exact frequency of each object class, the system should react faster (meaning that the system moves towards the decision faster) to feature values retrieved from the data and it would not be a fair comparison to the system using the hand-crafted networks. So a second test was performed where the prior beliefs for each object class in the networks were changed to the prior probabilities of the hand-crafted networks (System B). The results obtained for these two experiments are shown in Tables 6.1 and 6.2.

Table 6.1. Total number of calls to visual operators for the three basic datasets for all classes.

Decision process	Number of Operators
Hand-crafted Networks	430
Learned Networks (System A)	322
Learned + Adjusted Priors (System B)	400
All operators	906

Table 6.2. Summary of the recognition process for different data sets using the learned networks.

Hand-crafted Networks				
Data set	Overall	Level 0	Level 1	Level 2
Fort Hood	35/42	37/42	23/25	21/21
Avenches	13/18	16/18	12/13	5/7
Fort Benning	16/19	18/19	17/18	16/17
Total	64/79	71/79	52/56	42/45
Learned Networks - System A				
Data set	Overall	Level 0	Level 1	Level 2
Fort Hood	33/42	34/42	20/21	20/20
Avenches	16/18	18/18	15/15	7/9
Fort Benning	15/19	18/19	17/18	15/17
Total	64/79	70/79	52/54	42/46
Learned Networks + Adjusted Priors - System B				
Data set	Overall	Level 0	Level 1	Level 2
Fort Hood	34/42	35/42	20/21	20/20
Avenches	13/18	16/18	12/14	6/7
Fort Benning	16/19	18/19	17/18	16/17
Total	63/79	69/79	49/53	42/44

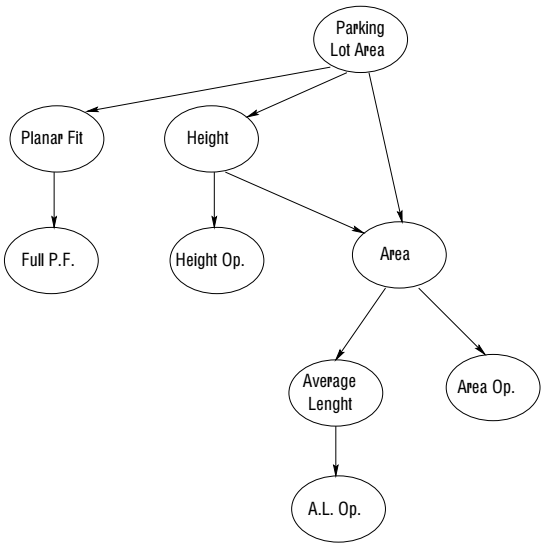

Figure 6.4. The level 1 network for parking lots learned from data determines if a region classified as a parking lot is a real parking lot, an RV-truck, or a single vehicle.

The numbers shown in Table 6.2 are all similar. Thus, the system using Bayesian networks learned from data performed very similar to the system using the hand-crafted networks in terms of number of regions correctly classified. However, System A was able to classify the regions using 32% fewer operators and System B used 15% fewer operators than the system using the hand-crafted networks (see Table 6.1). The fact that System B used more operators than System A was expected because the distributions of beliefs over the object classes were more uniformly distributed in System B than in System A, thus requiring more exploratory calls before deciding about a region. The fact that System B is more densely connected than the hand-crafted network explain the fact that it used less operators to get the same performance in terms of number of regions correctly classified as the hand-crafted network.

6.3.1 Using the learned networks on two new datasets

The next experiment was designed to show that the structure and relationships among features learned from data is robust enough to be applied to a different dataset. In this experiment, the hand-crafted system using utility theory was compared to the

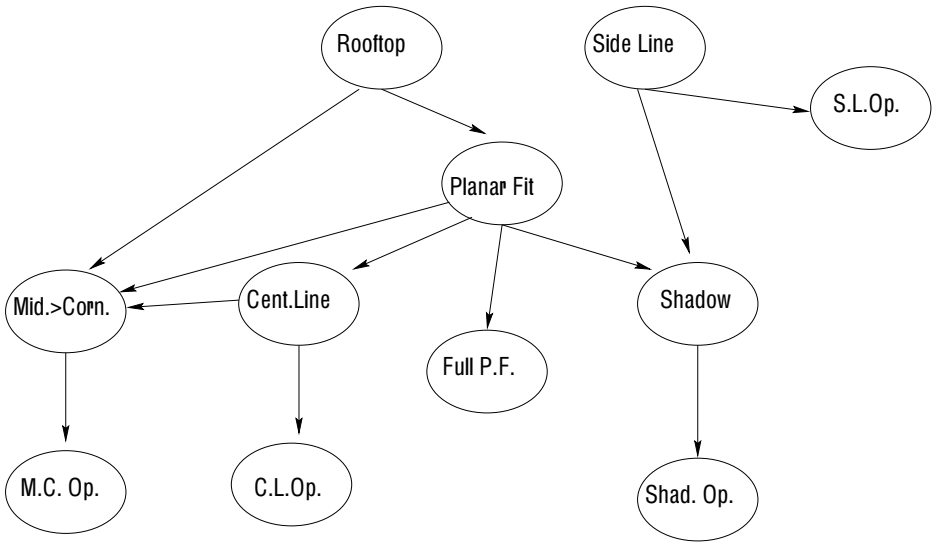

Figure 6.5. This level 2 learned network is called after a single building is detected. It is used to determine the building's rooftop type (Peak or Flat).

Table 6.3. Probability distribution of beliefs in the root node for the level 0 network among the states Building, Parking Lot, Open Field, and Other for the Flat Scene and Glandorf data sets.

Data Set	Building	Parking Lot	Open Field	Other
Flat Scene	0.42	0.21	0.27	0.1
Glandorf	0.50	0.30	0.12	0.08

learned system applied to a set of regions extracted manually from the Flat Scene dataset and from the Glandorf dataset, as shown in Figures 6.6 and 6.7. In both systems the prior beliefs were adjusted to approximate the distribution of object classes in each dataset (as shown in Tables 6.3 to 6.6). The results are shown in Tables 6.7 and 6.8, and Figures 6.8 and 6.9.

Table 6.4. Probability distribution of beliefs in the root node for the level 1 network for Buildings between the states Multilevel and Single Level for the Flat Scene and Glandorf data sets.

Data Set	Multilevel	Single Level
Flat Scene	0.35	0.65
Glandorf	0.35	0.65

Table 6.5. Probability distribution of beliefs in the root node for the level 1 network for Parking Lots among the states Parking Lot, Truck-RV, and Single Vehicle for the Flat Scene and Glandorf data sets.

Data Set	Parking Lot	Truck-RV	Single Vehicle
Flat Scene	0.23	0.23	0.54
Glandorf	0.33	0.12	0.55

Table 6.6. Probability distribution of beliefs in the root node for the level 2 network for Buildings between the states Flat Roof and Peak Roof for the Flat Scene and Glandorf data sets.

Data Set	Flat Roof	Peak Roof
Flat Scene	0.30	0.70
Glandorf	0.45	0.55

Table 6.7. Summary of the recognition process for the Flat Scene dataset using the hand-crafted and the learned networks with utility theory and adjusted priors.

Flat Data Set					
System	Overall	Level 0	Level 1	Level 2	Operators
Hand-crafted	22/30	23/30	21/21	13/14	170
Learned	26/30	27/30	21/21	13/14	162

Table 6.8. Summary of the recognition process for the Glandorf Scene dataset using the hand-crafted and the learned networks with utility theory and adjusted priors.

Glandorf Data Set					
System	Overall	Level 0	Level 1	Level 2	Operators
Hand-crafted	46/80	61/80	56/65	29/35	363
Learned	48/80	57/80	58/65	33/35	455

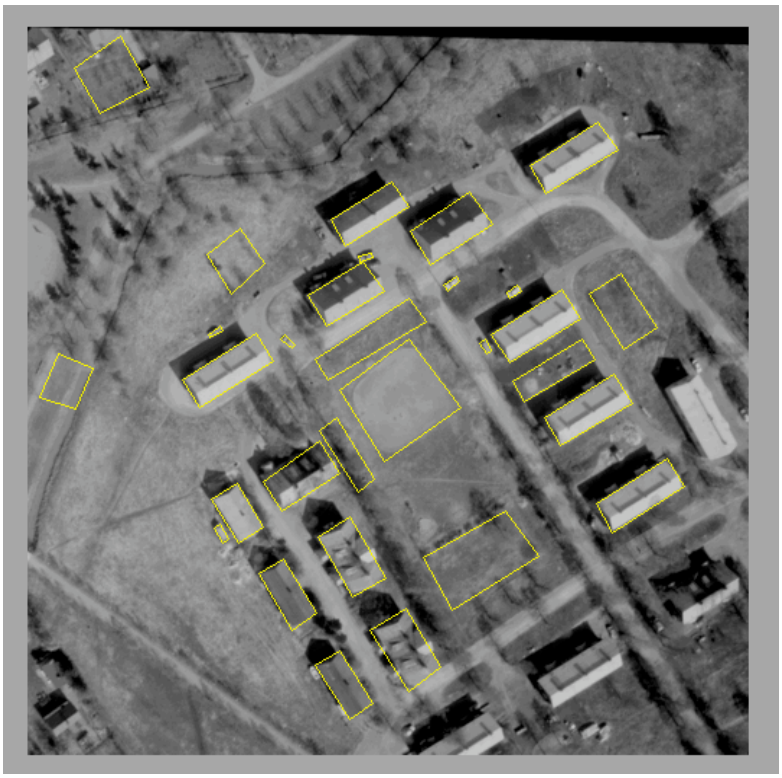

Figure 6.6. Set of regions extracted by hand from the Flat Scene dataset.

In the Flat Scene dataset the number of operators applied by the system using the learned networks is slightly smaller (5%), but the larger number of relationships between the features in the learned networks allowed a clearly better performance of the system (87% correct classifications against 73% for the system with the hand-crafted networks). An improvement in the performance of the system using the learned networks was also observed in the Glandorf dataset (60% against 57.5% for the system with the hand-crafted networks), but the number of operators used by the learned networks in this case was considerably higher than the system using the hand-crafted networks (25%). In the Glandorf dataset 10% of the regions incorrectly classified are complex regions that have parts of different objects in it. These regions could be classified as anything. Another problem with the Glandorf dataset is that the images are darker than the others, so the sequence of operators learned using the other datasets might not work as well in the presence of poor contrast (line detectors,

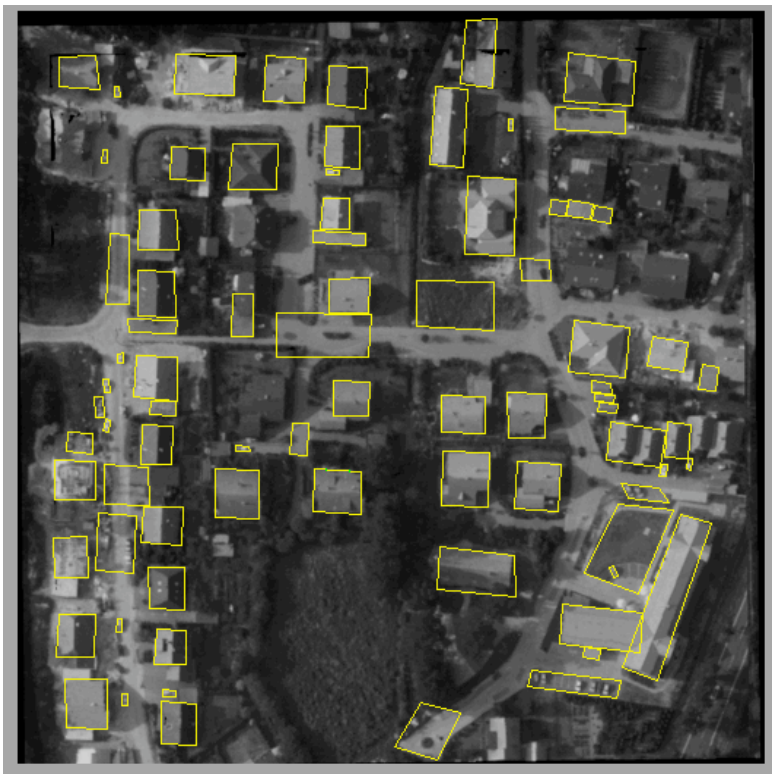

Figure 6.7. Set of regions extracted by hand from the Glandorf dataset.

for instance). Another problem in this data set is that some objects are very close to others, and when computing features from the DEM an overlap of the regions could lead to mistakes (height, for instance, compares the average height inside the region with the average height of the region's neighborhood). Finally, the set of prior probabilities used in this data set were not tuned. Other values for prior probabilities could give the system a better performance, but this is a time consuming task that is beyond the scope of this work.

One aspect to notice is that the learned networks have a higher density of links (as mentioned before), meaning that features are correlated among themselves (as expected). This correlation was not explored when the hand-crafted networks were designed (there was no practical way to determine the conditional probability tables between features at that point). In this case, evidence of the presence of one feature gives not only evidence about the object label but also evidence about the presence

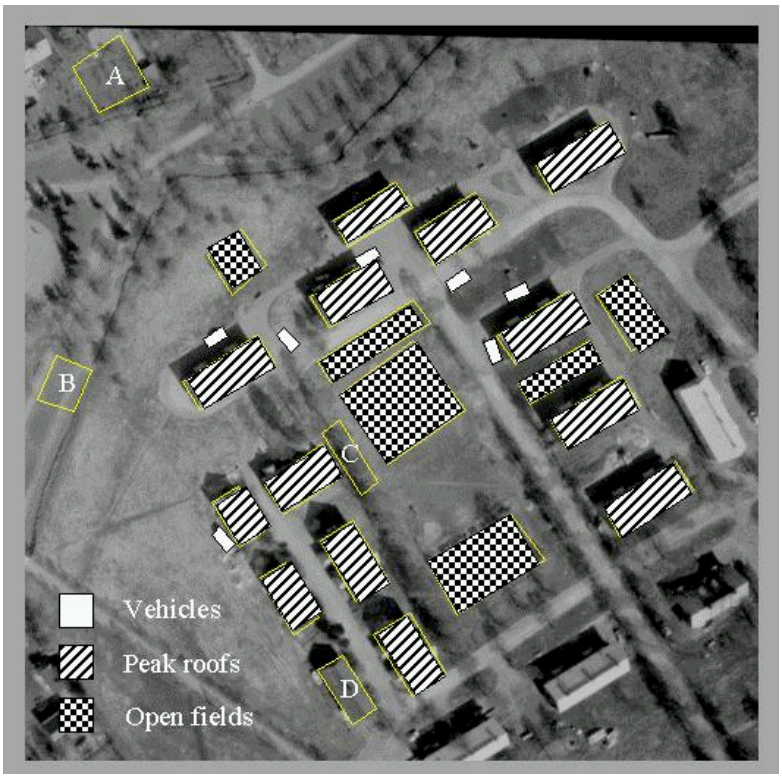

Figure 6.8. Classified regions for the Flat Scene dataset.

or absence of other features. This is one explanation of why the learned networks converged faster to the correct classification than the hand-crafted networks.

These two points will require an investigation on how the relations among features can improve the system's performance as suggested above. One must also consider that a higher density of links in the network implies a longer propagation time. A balance between link density and performance (in terms of correct classification, number of operators used, and processing time) has to be found for each network. This will be left to future work.

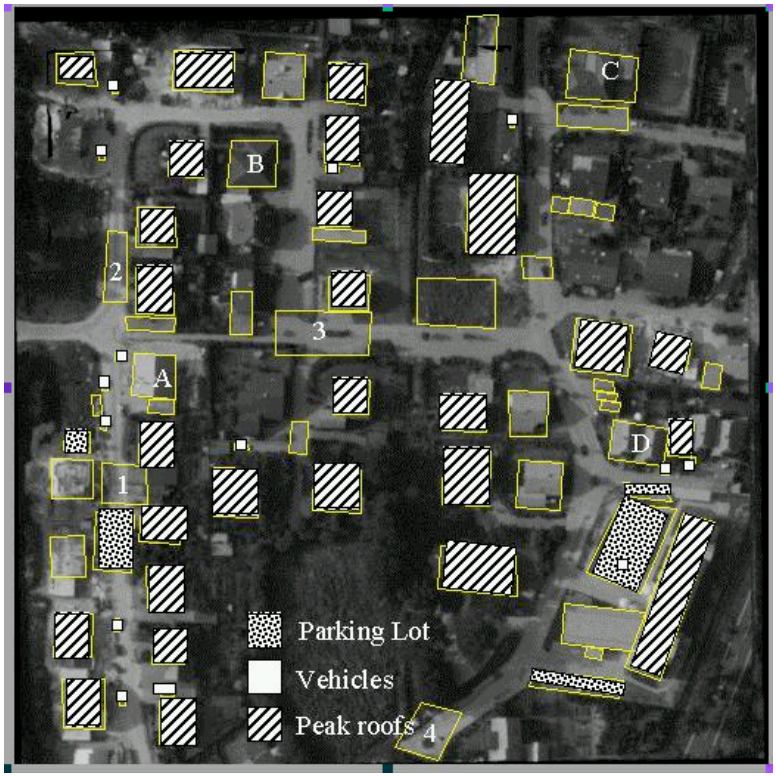

Figure 6.9. Classified regions for the Glandorf dataset.

CHAPTER 7

CONCLUSIONS

7.1 Conclusions

At the beginning of this thesis the following questions were asked:

1. How can the results of visual operators and their associated uncertainties be combined in order to classify a particular image region?

2. How can the hierarchical structure of objects be exploited in order to construct an incremental classification process?

3. Can the construction of the knowledge base be simplified (or fully automated) for a particular application using both human expertise and machine learning techniques?

4. Can performance be improved by using a disciplined approach to operator selection?

We have shown that Bayesian networks are robust and they can be used to combine converging and conflicting information coming from different features. The uncertainty about features measurements and the relations between features and object classes is mapped into the conditional probability tables. In the data sets used here the Ascender II system using Bayesian networks correctly classified regions in aerial images with accuracies above 80% on four of the five datests, and about 60% on the other.

The hierarchical structure of Bayesian networks used in the knowledge base allowed the system to work incrementally through partitioned sub-networks. The regions are classified first using a more generic label, which is refined in the subsequent levels. The drawback related to backtracking was never really an issue in the experiments developed here, since the few cases where it was needed could be anticipated and corrective strategies were incorporated as necessary.

We have demonstrated that visual information provided by humans in a simple and natural way can be combined and used to learn the structure and/or conditional probability tables for the Bayesian networks. The data used to learn the networks must be representative of all object classes whose recognition by the system is desired. These networks can be used on similar datasets with about the same performance as a hand-crafted system with a simple adjustment of prior beliefs for the object classes. This significantly reduces the burden of designing and tuning the structure and probability tables for Bayesian networks, which has been a major criticism of this methodology.

A more disciplined approach to operator selection and decision making such as utility theory proved to be more efficient, not in terms of number of regions correctly classified, but in terms of resources used. In general the system using utility theory used fewer operators to classify regions than the system using uncertainty distance. When operator cost was added to the system, resource requirements were further reduced. Utility theory also allow a simpler way to set personal preferences into the system or to change the system's goals.

In the Ascender II system the knowledge base and control processes (reasoning subsystem) are completely separated from the visual subsystem (visual operators, models, images). The separation allows more flexibility in adding new visual operators, or in replacing current operators, and requires only minor changes in the reasoning subsystem.

The datasets used with the Ascender II system differ in the types of buildings present, image resolution, camera models, number of views available, imaging conditions, etc, but the system correctly identified most of the regions in each site without any adjustement. The prior probabilities in the root node of the Bayesian network were changed for each site just to reflect the distribution among object classes in each dataset.

In addition to these specific properties, the Ascender II system has some features that are desirable in a general vision system, as discussed in the surveys [19] and [22].

- The control of visual operators was addressed and we show an efficient methodology for operator selection and classification.
- The use of Bayesian networks and the separation of the reasoning subsystem from the visual subsystem facilitates the addition of new features to the knowledge base and the addition of new operators to the visual subsystem's library.
- The system has multiple levels of representation for the object classes provided by the hierarchical structure of Bayesian networks.
- The selection of operators is performed dynamically and depends on the system's current state of knowledge.
- Conflict resolution, evidence combination and mapping of numerical values into symbolic quantities are all provided by the Bayes nets framework in a principled manner.

7.2 Future Work

The Ascender II system is an ongoing project. It is possible to add more operators and create more branches to include new object classes. The control strategies will be refined to improve performance, either in terms of processing time, or in terms of

correct classification. There is a large set of research topics that could be pursued within the current framework. Some of the more interesting questions are presented below.

7.2.1 Related to Vision

1. The current system has only one visual operator associated with most features. It is possible to add more operators per feature, and to decide which operator to use once a feature is selected. The idea would be to extend the feature selection process currently used to select the operator which will measure the feature.

2. The system could be used to simulate new features and the visual operator(s) for extracting it before implementation of the feature. This simulation would help to identify characteristics of the feature/visual operator (e.g., reliability) that would be useful in the classification process and to determine the impact of the feature on the system's performance. This was done with the shadow operator in the current system. The operator was not implemented. The use of the shadow feature was simulated assuming an operator reliability of 80%. A conditional probability table reflecting this reliability was added to the Bayesian network at level 2 and a set of experiments were performed. The experiments show that an operator with this reliability would improve classification. As only one operator was tested under this methodology, further study is required to fully understand the simulation aspects of the system.

3. The current system could be expanded to incorporate local spatial relations between objects in the scene (e.g., parking lots nearby roads).

4. The current system could also be expanded to perform temporal reasoning. Suppose that a certain scene was reconstructed a year ago and a new image is obtained from the same area. How could this previous information (i.e. prior

reconstruction) be used to drive the system to find changes efficiently in the new scene?

7.2.2 Related to Bayesian Networks

1. We already used different prior probabilities in each network and verified that the system's behavior changes significantly. A more detailed study should be done to show how changes in the prior probabilities affect the overall system's performance, both in terms of processing time and in terms of correct classifications.

2. We know that certain features extracted from individual regions are related, like *ratio, width and area* in the level 0 network (in this case we simply added a link between them). However, it is not clear how some features are related to others, for instance, is *planar fit* related to *height*? We also know that the number of connections will affect the propagation time, thus affecting the overall processing time. The learned networks show that the structure of relations among features can improve the system's performance, but a more detailed study about how features are connected and how and why performance varies as a function of feature links is needed.

3. The number of features modeled in the networks is directly related to the number of nodes present in the networks. The tradeoff between features, network complexity and processing time is also an open question.

7.2.3 Related to Anytime Systems

An "anytime" system is defined as a system designed to improve the quality of its computation continuously over time [60]. The Ascender II system, because of its hierarchical structure, could be modified to show an anytime system behavior. This could be done in two different non exclusive ways: within each level (the selection of

an operator increases the belief of the region's outcome over time), and within the hierarchical structure (the refinement of the classification process can be seen as an improvement in terms of detail in the region's final label).

One way to show improvement in the Ascender II system is to measure the system's utility over time. Because algorithm improvement is not done continuously, in most of the practical anytime systems the improvement is not observed as a continuous function, but it is more like a step function. In the Ascender II system this would also be the case because there is no improvement when an operator is being applied. The improvement has to be measured before and/or after the application of an operator that requires a discrete interval of time.

One problem in demonstrating the Ascender II system as an anytime system is that the average number of operators applied per level is too small (less than 2), so the step function would have only three points. The first issue we have to deal with in this problem is to define a function to combine the different levels of reasoning as we outline below.

Consider that the first level is the most important one since the system defines the region's most general object class in the first level, and after that it is limited to refining its classification. We could define a function to weight the system's decision over the different levels of reasoning as shown below:

$$Level_{weight} = decision_0 + 0.1 * decision_1 + 0.1 * decision_2$$

Suppose $Region_i$ is a *Parking Lot* and the prior probabilities show a bias for *Buildings*. Without using any operator the decision at level 0 would be for *Building*, which would be wrong, so we have that the $decision_0$ without any operator is 0 (a wrong decision). After applying the first selected operator the prior beliefs change. Suppose now that the highest belief is for *Parking lot*. The $decision_0$ after the first

operator is now 1 (the correct decision). This process can be repeated through all levels such that each region would get a vector of decision values after each operator.

Using the equation above it is possible to plot a curve of $Level_{weight}$ versus the number of operators used. Averaging over the curves for all regions, we get the graph presented in Figure 7.1.

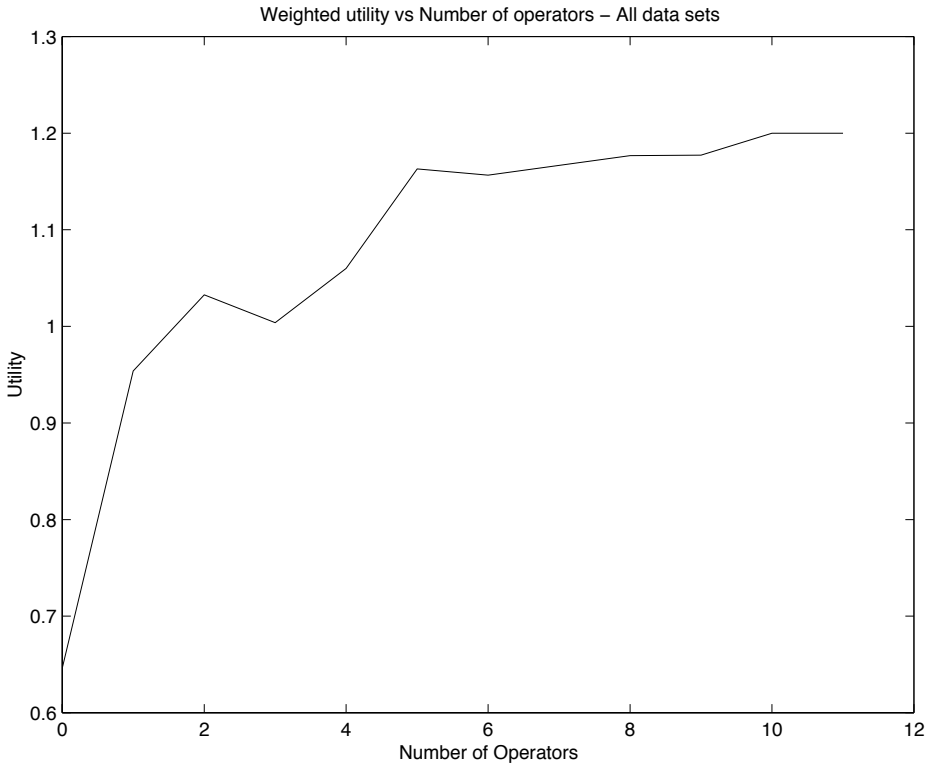

Figure 7.1. Average weighted utility in the Ascender II system. It shows how the overall utility is generally an increasing function. Thus it could behave as an anytime system

The problem in our case is that the number of regions in each level was not large enough, otherwise the graph should give the average utility of an operator in the sequence, which should be an increasing continuous function. A more detailed study has to be carried out to understand the system's behavior as an anytime system.

APPENDIX A

INTRODUCTION TO BAYESIAN NETWORKS

This appendix is included for readers not familiar with the terminology of Bayesian Networks. The appendix presents the basic terminology, some concepts about the network structure, and some ideas about how to use a Bayesian network. A knowledgeable reader can safely skip it.

A.1 The Bayesian Network Structure

A Bayesian network is a Directed Acyclic Graph (DAG). Each node in the network represents a Random variable and each edge represents a relationship between two variables. Each variable is either continuous or discretized into a set of exhaustive mutual exclusive states. The Bayesian Network has two types of nodes, an external node and an internal node. An external node has no parents and an internal node has at least one parent.

All nodes have a probability table associated with it. The table for a variable in an external node have the prior probabilities for each possible state of the variable. The table for a variable in an internal node represents the conditional probability associated with that variable. The conditional probability table has values such as $P(X = x|Y_1 = y_1, Y_2 = y_2, \cdots, Y_n = y_n)$, where X is the variable represented by the internal node and Y_i is the set parents of node X.

Figure A.1 shows an example of a simple Bayesian Network. All variables in this network are Boolean with states true and false. Nodes A and E are external nodes (or roots). Nodes B, C, and D are internal nodes. The prior probability tables for

nodes A and E give the values of $P(A = True)$ and $P(A = False)$ and $P(E = True)$ and $P(E = False)$, respectively. The other 3 tables give the conditional probability values for $P(B|A)$, $P(C|A)$, and $P(D|C, E)$.

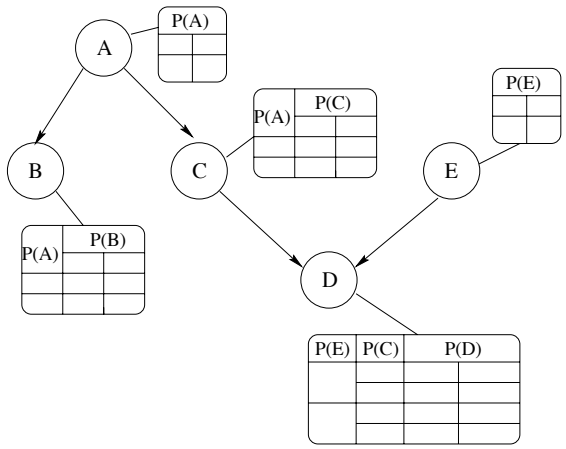

Figure A.1. Simple Bayesian network with corresponding probability tables.

One of the advantages on using Bayesian Networks is that it facilitates the computation of joint probabilities, that is, suppose we want to know the probability of the joint event: $P(D = True, C = False, A = False)$ or simply $P(D, \neg C, \neg A)$.

In terms of probability theory, when dependencies are not clearly stated, this is computed by:

$$P(D|C, A) * P(C|A) * P(A)$$

but in our Bayesian Network example this is computed as:

$$P(D|C) * P(C|A) * P(A)$$

because D is only directly dependent of C.

Bayesian Networks can be used to make different kinds of inferences, as presented in Russell [54]:

- Diagnostic: from effects to cause (given A infer about D).

- Causal: from causes to effects (given D infer about A, C, and E).
- Intercausal: between causes of a common effect. This is also known as explaining away. (given D and C we can explain what happened with E).
- Mixed: combining two or more of the above.

A.2 Forecast example using Bayesian networks

This simple example is extracted from Russell [54] and will be used to show the reasoning process using Bayesian Networks.

A.2.1 Building a Model

Consider the following situation: one morning Mr Holmes realizes that the grass in his front yard is wet. He is puzzled. He wants to know the cause for the wet grass. There are two possibilities: either rain occurred last night or he forgot to turn off the sprinkler last night. Mr Holmes has a neighbor (Dr Watson) whose house also has a lawn. He knows that if the lawn in Dr Watson house is wet it is more likely that rain caused the wet grass.

The situation described above can be modeled as a Bayesian Network. Notice that there are four variables described in the text: Mr Holmes lawn, Dr Watson lawn (both have only two states wet or dry), Rain occured last night, and the Sprinkler was on (these two also have two states: yes or no).

As Mr Holmes lawn can be wet either by rain or by the sprinkler, and as we know nothing about Dr Watson sprinkler, we can built the Bayesian Network presented in Figure A.2.

Now, Mr Holmes knows that 10% of the time he forgets to turn off the sprinkler and he also knows that 20% of the time rains at night. Based on these information it is possible to build the probability tables for both root nodes, as shown in Tables A.1 and A.2.

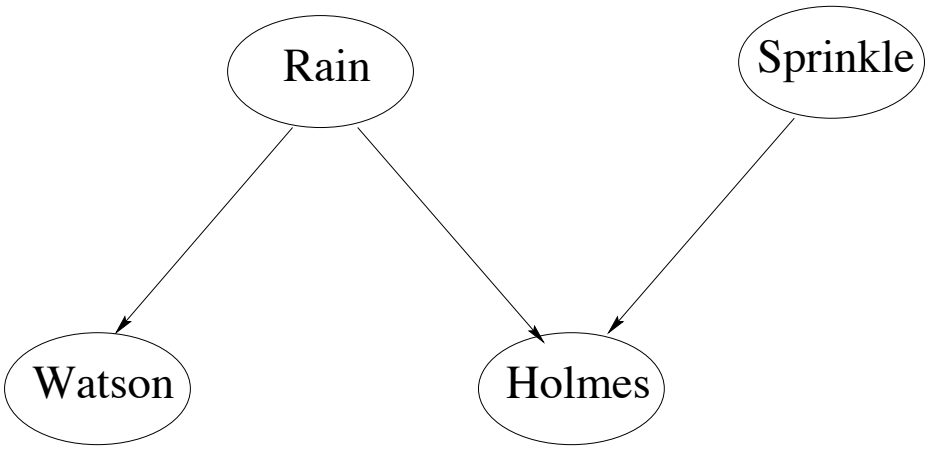

Figure A.2. Simple Bayesian network.

Table A.1. The prior probability table for rain.

	YES	NO
P(Rain occurred)	0.2	0.8

To build the conditional probability tables we need to know how the variables are related. Lets consider first Dr Watson lawn condition: if rain occurred last night Dr Watson lawn is certainly wet, with probability one. If rain did not occurred last night Dr Watson lawn could be wet because he also forgot his sprinkler on, lets consider (or simply make a guess) that Dr Watson lawn is wet 20% of the time when there is no rain. Notice that in order to be straight we should give the probability of the grass being wet without rain as 50% - we know nothing about - but, as we know Mr Holmes behavior forgetting the sprinkler on we can make a better guess for this probability. This situation leads to the conditional probability table presented in Table A.3.

Reasoning similarly about Mr Holmes lawn we can built the conditional probability table presented in Table A.4. Notice that if rain occurred Mr Holmes lawn will

Table A.2. The prior probability table for the sprinkler on.

	YES	NO
P(Sprinkler on)	0.1	0.9

Table A.3. The conditional probability table of the grass in Dr Watson's house being wet given the knowledge of rain.

	P(Rain occurred=yes)	P(Rain occurred=no)
P(Watson's lawn=wet)	1	0.2
P(Watson's lawn=dry)	0	0.8

Table A.4. The conditional probability table of the grass in Mr Holmes' house being wet given the knowledge of rain and the sprinkler on.

P(Rain)	P(Sprinkler)	P(Lawn=wet)	P(Lawn=dry)
yes	yes	1	0
yes	no	1	0
no	yes	0.9	0.1
no	no	0	1

be certainly wet, but if the sprinkler was on there is a possibility (10% of the time) that the lawn dries at night and it is not wet in the morning.

The whole network can be broken into two clusters (Rain, Holmes, Sprinkle, and Rain, Watson), because $Watson$ is independent of $Holmes$ (if we don't know about $Rain$ then the condition of Holmes' front yard is not dependent of the condition of Watson's front yard). The usage of this clusters of variables is the basic idea behind the HUGIN system [2]).

A.2.2 Answering questions without evidence

Once the model is built it is possible to determine the expected events. At first we know that Mr Holmes was puzzled that the lawn is wet, the question is: what is the overall probability that Mr Holmes lawn is wet? To answer this question we have to compute $P(MrHolmesLawn = wet)$ and $P(MrHolmesLawn = dry)$ (in this case having one of them is enough because the variable has only two states and their probabilities add up to 1). To compute this probabilities we have to analyse all possible combinations of $P(RainOccurred)$ and $P(SprinklerOn)$. This is given by

Table A.5. The joint probability table for Mr Holmes lawn, Rain occurred and Sprinkler on.

P(Rain)	P(Sprinkler)	P(Lawn=wet)	P(Lawn=dry)
yes	yes	0.02	0
yes	no	0.18	0
no	yes	0.072	0.008
no	no	0	0.72

the joint probability $P(MrHolmesLawn, RainOccurred, SprinklerOn)$, all together, which is given by the expression below:

$$P(Holmes, Rain, Sprinkler) = P(Holmes|Rain, Sprinkler) * P(Rain, Sprinkler)$$

but as *Rain* and *Sprinkler* are independent we have:

$$P(Holmes, Rain, Sprinkler) = P(Holmes|Rain, Sprinkler) * P(Rain) * P(Sprinkler)$$

The result of the joint probability is presented in Table A.5, $P(MrHolmesLawn, RainOccurred, SprinklerOn)$, which shows that the event more likely is that no rain occurred, the sprinkler was off and Mr Holmes lawn was dry. By marginalization[1], we can compute $P(MrHolmesLawn)$ which is presented in Table A.6 which gives that 72.8% of the time Mr Holmes lawn is expected to be dry and 27.2% of the time Mr Holmes lawn is expected to be wet. As the lawn is expected to be dry more often this explains the reason that Mr Holmes was puzzled. The process can be repeated for $P(DrWatsonLawn)$ and $P(RainOccurred)$ and we get Tables A.7 and A.8.

A.2.3 Handling evidence and answering other questions

A Bayesian Network is a model that has a set of variables and a set of relations. The variables usually represent an event (e.g. Rain occurred, Mr Holmes lawn).

[1]Addition of each column in the table

Table A.6. Probability of the grass in Mr Holmes' house being wet.

	WET	DRY
P(Lawn)	0.272	0.728

Table A.7. The joint probability table for Rain occurred and Dr Watson lawn being wet.

	P(Rain occurred=yes)	P(Rain occurred=no)
P(Dr Watson lawn=wet)	0.2	0.16
P(Dr Watson lawn=dry)	0	0.64

When an event is observed we say that evidence was gathered for that event (e.g. Rain occurred = yes, Mr Holmes lawn = dry).

When one or more events are observed and their values are set in the corresponding variables and propagated through the model the expected values for other variables are changed. Using this technique it is possible to explain some events or change the expectation related to some unobserved events.

Now lets see how we handle evidence in this example. Lets suppose we have the evidence that the grass in Mr Holmes front yard is wet $P(MrHolmesLawn = wet)$. The first thing we do is to set the probability $P(MrHolmesLawn = dry)$ to zero in Table A.5, as presented in Table A.9:

Note that the values now do not add up to 1, so we have to normalize these probabilities by dividing them by the prior $P(MrHolmesLawn = wet) = 0.272$ (see Table A.10). This table represents the posterior joint probability $P^*(MrHolmesLawn, RainOccurred, SprinklerOn)$. By marginalization we can also compute the poste-

Table A.8. Probability of Dr Watson lawn being wet and dry.

	WET	DRY
P(Dr Watson lawn)	0.36	0.64

Table A.9. Table for the joint variables Rain occurred, Sprinkler on and Mr Holmes lawn when evidence about the grass being wet is entered.

P(Rain)	P(Sprinkler)	P(Lawn=wet)	P(Lawn=dry)
yes	yes	0.02	0
yes	no	0.18	0
no	yes	0.072	0
no	no	0	0

Table A.10. Normalized table for the joint variables Rain occurred, Sprinkler on and Mr Holmes lawn when evidence about the grass being wet is entered.

P(Rain)	P(Sprinkler)	P(Lawn=wet)	P(Lawn=dry)
yes	yes	0.074	0
yes	no	0.662	0
no	yes	0.264	0
no	no	0	0

rior probabilities for $P^*(RainOccurred)$ and $P^*(SprinklerOn)$, which are shown in Tables A.11, and A.12.

Notice that the probabilities of $P(RainOccurred = YES)$ and $P(SprinklerOn = YES)$ both raised considerably. This was expected since one of them or both were responsible for the fact that Mr Holmes lawn was wet. Using the posterior probability of Rain occurred it is possible to update the expectation for Dr Watson lawn. This is done by first computing the posterior joint probability $P^*(RainOccurred, DrWatsonLawn)$. This computation is performed by multiplying the Table A.7 by the posterior distribution of Rain, $P^*(Rain)$, (Table A.11) and dividing the resulting table by the prior distribution of rain, (Table A.1), that is:

Table A.11. Posterior probabilities for Rain occurred given that Mr. Holmes lawn is wet.

	YES	NO
P(Rain occurred)	0.736	0.264

Table A.12. Posterior probability for the Sprinkler on given that Mr. Holmes lawn is wet.

	YES	NO
P(Sprinkler on)	0.338	0.662

Table A.13. Posterior probability table for the variables Rain occurred and Dr Watson lawn being wet.

	P(Rain occurred=yes)	P(Rain occurred=no)
P(Dr Watson lawn=wet)	0.736	0.0528
P(Dr Watson lawn=dry)	0	0.2112

$$P^*(Watson, Rain) = P(Watson|Rain)P^*(Rain) = P(Watson, Rain)\frac{P^*(Rain)}{P(Rain)}$$

The values of the posterior probabilities $P^*(DrWatsonLawn, RainOcurred)$ are presented in Table A.13, and by marginalization we can compute the posterior probability table for Dr Watson lawn as presented in Table A.14.

Notice that once we know that Mr Holmes lawn is wet the probability of Dr Watson lawn being wet raised more than twice its initial value. Suppose now that Mr. Holmes checks Dr. Watson's front yard and sees that it is not wet, $(DrWatsonLawn = dry)$. The process is repeated by setting evidence in Table A.13, as shown in Table A.15.

By marginalization we can compute the new posterior probability of Rain occurred, $P^{**}(RainOccurred = no) = 1$, and we can update the posterior joint probability $P^*(MrHolmesLawn[H], RainOccurred[R], SprinklerOn[S])$ as follows:

$$P^{**}(H, R, S) = P^*(H, R, S)\frac{P^{**}(R)}{P^*(R)}$$

Table A.14. Posterior probability table for Dr Watson lawn.

	WET	DRY
P(Dr Watson lawn)	0.7888	0.2112

Table A.15. Probability table showing evidence that Dr Watson lawn is not wet.

	P(Rain occurred=yes)	P(Rain occurred=no)
P(Dr Watson lawn=wet)	0	0
P(Dr Watson lawn=dry)	0	1

Table A.16. The new posterior joint probability for Rain occurred, Sprinkler on and Mr Holmes lawn.

P(Rain)	P(Sprinkler)	P(Lawn=wet)	P(Lawn=dry)
yes	yes	0	0
yes	no	0	0
no	yes	1	0
no	no	0	0

The new posterior probability for Mr Holmes lawn, Rain occurred and Sprinkler on can be computed, as shown in Table A.16, which gives the event that the Sprinkler was on as certain, $P^{**}(SprinklerOn = yes) = 1$, which is the only possible explanation for Mr Holmes lawn being wet and Dr Watson lawn being dry.

Notice that if Mr Holmes found that the grass in Dr. Watson front yard was wet ($DrWatsonLawn = wet$) then we would have a more interesting propagation process. In this case evidence for the wet grass in Dr Watson lawn would be entered in Table A.13 and we would get Table A.17.

This fact would lead to a new posterior probability for Rain occurred $P^{**}(Rain\ Occurred = yes) = 0.933$ and a new posterior joint probability for the variables Rain occurred, Sprinkler on and Mr Holmes lawn being wet $P^{**}(MrHolmesLawn, RainOccurred, SprinklerOn)$, as shown in Table A.18. Notice that in this case

Table A.17. Probability table when evidence is entered saying that Dr Watson lawn is not wet.

	P(Rain occurred=yes)	P(Rain occurred=no)
P(Dr Watson lawn=wet)	0.933	0.067
P(Dr Watson lawn=dry)	0	0

Table A.18. Posterior joint probability table for the variables Rain occurred, Sprinkler on and Mr Holmes lawn being wet given evidence that Dr Watson lawn is wet.

P(Rain)	P(Sprinkler)	P(Lawn=wet)	P(Lawn=dry)
yes	yes	0.094	0
yes	no	0.839	0
no	yes	0.067	0
no	no	0	0

both events are possible, by marginalization $P^{**}(RainOccurred = yes) = 0.933$, and $P^{**}(SprinklerOn = yes) = 0.161$.

APPENDIX B

A DECISION MAKING EXAMPLE USING UTILITY THEORY

This appendix is included for readers not familiar with the terminology of Utility Theory. The appendix presents the basic terminology, some concepts and ideas about how to use Utility Theory. A knowledgeable reader can safely skip it.

B.1 The concept of utility and utility theory

We have to make decisions every day of our lives. Some of these decisions are simple such as what shirt to wear or where to go for lunch. A few of these decisions are more important such as a marriage proposal, an offer for a new job, or buying a new car. In certain cases these decisions reflect directly only in ourselves or some direct relatives, in other cases it might affect others (managers in industry usually make decisions that affect shareholders, employees and the community, government people can make decisions which can affect the whole nation and sometimes other nations). So decision making and its consequences is something that concerns all of us, both as makers of the action and/or as sufferers from the consequences.

When we make a decision we basically decide to take an action in a certain direction. In order to make the decision we usually consider some situations we encountered in the past, the circunstances we have in the present and the possible outcome for a set of events that might happen in the future. Each action taken carries a set of consequences depending on the outcome of these events in the future (for instance if we decide to buy a certain stock its value in the future depends on a set of things involving the market, the economy, etc).

On evaluating the consequences of possible actions the problem is to give a numerical value for each consequence. An easier task is to evaluate two possible consequences and decide which one is preferable. Using this concept and assuming that preference is transitive (i.e. if consequence A is preferable than consequence B and consequence B is preferable than consequence C, than consequence A is preferable than consequence C), we can give a partial order to all possible consequences [11] of taking an action. Once this partial order is achieved we can assign numbers indicating how much each consequence is valued. These numbers are called **utilities**, and **utility theory** deals with the development of such numbers [40] and how they can be used in a decision problem [44].

B.2 Building a model for Utility Theory

In this section we will be using a simple decision problem example to show how to build a model using utility theory, and how the model can help on a decision making process.

B.2.1 Buy a used car example

This example shows how utility theory can be used in a decision making problem. Consider the following decision problem: we are considering buying a specific used car. Before we buy the car we can perform a test to know about the car's condition. Let's consider the case where two tests are available: *Test 1* is 100% reliable, expensive but gives a good profile about the car, and *Test 2* is 100% reliable, it is cheaper, but reports only the car's current condition. The possible unknown events are related to the car's current condition which can be either good or bad. The decisions we are facing are: buy the car without performing any test, perform *Test 1* on the car and buy the car if it passes the test, or perform *Test 2* on the car and buy the car if it passes the test. The consequence table for this example is presented in Table B.1.

Table B.1. Consequence table for the "buy a used car" problem.

Decisions	*Events*	
	Car=Good	Car=Bad
d_1=Perform no Test	C_{11}	C_{12}
d_2=Perform Test 1	C_{21}	C_{22}
d_3=Perform Test 2	C_{31}	C_{32}

The best consequence in the first column is C_{11}, because if the car is good we can just buy it without performing any test. The worst consequence in the first column is C_{21} because *Test 1* is the most expensive test available, and if the car is good we will spend the largest amount of money to obtain that information. In the second column the best consequence is C_{32}, because if the car is bad we spent the least amount of money to determine the car's condition. The worst consequence in the second column would be C_{12}, that is, buy the car if the car is bad. All the consequences can be compared and partially ordered. One possible partial ordering is shown below:

$$C_{11}, C_{31}, C_{32}, C_{21}, C_{22}, C_{12}$$

At this point numerical values have to be defined for the consequences C_{ij}. We start this task by first giving values for the best and worst consequences in the list above, that is C_{11} and C_{12}. Without loss of generality assume the values 1 and 0 for the consequences C_{11} and C_{12} respectively. The next step consists in converting all other consequences into a numerical value. The numerical values, or utilities, are computed as follows:

For all consequences C_{ij} in the table find the corresponding value of p_{ij} as follows:

- Choose any consequence C_{ij}. C_{ij} is worse, or at most as good as the best consequence; and C_{ij} is better, or at least no worse than the worst consequence.
- Define the probability p_{ij} using the following methodology: suppose you are given a choice. You can have consequence C_{ij} with certainty, or you can gamble

to have consequence C_{11} with probability p_{ij} and C_{12} with probability $1 - p_{ij}$. For instance consider consequence C_{31} (Perform *Test 2* and buy the car), now consider this consequence as a fact, and consider the following lottery: there is a probability (p_{ij}) that the car is good and we can buy it without any test and there is another probability ($1 - p_{ij}$) that the car is bad and we are not buying it. If $p_{ij} = 1$ you will select to gamble, and if $p_{ij} = 0$ you will select C_{ij}. So find a value of p_{ij} that makes the selection between C_{ij} and the gamble indifferent.

The value of consequence C_{ij} is computed using the equation presented below:

$$C_{ij} = p_{ij} * C_{11} + (1 - p_{ij}) * C_{12}$$

The set of probabilities p_{ij} can be computed for each consequence C_{ij} as explained above. Table B.2 shows possible values for each probability p_{ij} (because of the subjectivity involved, these values could be different for another person). Notice that because $C_{12} = 0$ and $C_{11} = 1$ the values of C_{ij} (utilities) are equal the probabilities p_{ij}.

Another way to explain the utilities of the consequences in Table B.2 is the following: the values 1 and 0 do not need further explanation. The other entries can be viewed as follows: in the *Perform Test 2* case, if the car passes the test, the consequence is basically the same, the car is good and we will buy it, but we have to pay the car's test, which decreases the utility value from 1 to 0.8. If the car fails the test, we will not buy it, but we have to pay for the test and we will have no car at the end, which decreases the utility value a bit more from 1 to 0.7. Similarly if we decide to *Perform Test 1* and the car passes the test we will buy it but we spent the largest amount of money, decreasing its utility from 1 to 0.6. In this case, if the car fails the test we will not buy it but we spent the largest amount of money, so the utility value, in this case, decreases from 1 to 0.5.

Table B.2. Probability and , in this case utility table for the "buy a used car" problem.

Decisions	*Events*	
	Car = Good	Car = Bad
d_1=Perform no Test	1	0
d_2=Perform Test 1	0.6	0.5
d_3=Perform Test 2	0.8	0.7

Table B.3. Decision table for the "buy a used car" problem.

Decisions	*Class*	
	Car = Good	Car = Bad
d_1=Perform no Test	1	0
d_2=Perform Test 1	0.6	0.5
d_3=Perform Test 2	0.8	0.7
Probability	p(Good)=0.7	p(Bad)=0.3

The decision table for the "buy a used car" example is shown in Table B.3. The probabilities of the events "Car is Good" and "Car is Bad", in this case, could be obtained through the experience of the person analyzing the car's overall condition.

Using decision table B.3, it is possible to compute the utility of each decision using the equation presented below:

$$EU(d_i) = \sum_{j=1}^{n} p(event_j) * C_{ij}$$

That is, for each decision we make the summation for all possible events (n) of the product of the probability of the event and the utility of its consequence. The result of this sum is shown in Table B.4. The best decision is the one with the highest expected value, in this case the decision with highest utility is to *Perform Test 2* (with value 0.77). So the person should perform the test and buy the car if the car passes the test

Table B.4. Expected utility for each decision in the "buy a used car" problem.

Decisions	*Utility*
d_1=Buy the car	0.7
d_2=Perform Test 1	0.57
d_3=Perform Test 2	0.77

B.2.2 Dealing with uncertainty

In the previous example we assumed that the outcome of both tests were 100% reliable, this is almost never true. Information, in general, has a certain amount of uncertainty. Let's change the previous example and consider the following situation: We want to buy a car but we also want to perform a test to know the car's condition, but, in this case, we will consider that the tests are not 100% reliable but we have to select one of them or none. This problem is also known as the value of information, that is, how much should we pay for an information knowing that it is not completely reliable? Or, is a certain information worth a value given its reliability?

In this example we have only to select what test to perform on the car. We know that *Test 2* is 90% reliable and depends on the car's condition and the technician who will perform the test, but that the test does not give any information other than the car's current condition. *Test 1* gives a good profile about the car, but it needs some electronic equipment and the test's outcome depends on the quality of the electronic equipment being used. Assume that the electronic devices are good 3 out of 4 times.

The utility table in this problem is considerably different. We want to perform a test and use the information given by the test to decide if we will buy the car or not. If the car is good we want to buy it, but if the car is not good we don't want to buy it. The decision table for the buy a used car problem is shown in Table B.5. In this example there are two related decisions, the first decision is: what test to perform. The second decision is about buying or not buying the car, which depends on the outcome of the test performed.

Table B.5. Utility table for the buy a used car problem with a test selection.

Decisions	*Car's Condition*	
	Car=Good	Car=Bad
d_1=Buy the car	1	0
d_2=Do not buy the car	0	1

Table B.6. Conditional probability table for the Test 1 case.

Test outcome	*Car's Condition*	
	Car=Good	Car=Bad
Good	0.75	0.25
Bad	0.25	0.75

To compute the utility of each test and decide which one is the best test we will represent the reliability for each test as a conditional probability table, as shown in tables B.6 and B.7, for tests 1 and 2 respectively. The prior probabilities about the car's current condition are also known (expectations about the car's condition) (see Table B.8).

Now we can put all this information together and decide which test to select in order to get information about the car's condition and then decide whether to buy the car or not. In this case we have to compute the expected utility for each test and select the test with the highest expected utility. The expected utility for each test is computed using the expression:

$$EU(Test) = \sum_{States-of-Test} max_i \sum_{j=1}^{N} u(C_{ij}) p(Test = k/Car = j) p(j)$$

Table B.7. Conditional probability table for the Test 2 case.

Test outcome	*Car's Condition*	
	Car=Good	Car=Bad
Good	0.9	0.1
Bad	0.1	0.9

Table B.8. Prior probabilities for the car's current condition.

	Car's Condition	
	Good	Bad
Probability(Car=)	0.7	0.3

Notice that, because of Bayes Rule the equation above can also be written as:

$$EU(Test) = \sum_{States-of-Test} max_i \sum_{j=1}^{N} u(C_{ij}) p(Car = j/Test = k) p(Test = k)$$

This last expression will be used in the Ascender II system for the feature's expected utility computation. For the Test 1 case, the expected utility computation is summarized in Table B.9. The values for each entry in the Test 1 Outcomes were computed using the second summation in the formula:

$$\sum_{j=1}^{N} u(C_{ij}) p(Test = k/Car = j) p(j)$$

In the expression above N is the number of possible outcomes for the event (in the example, the car's condition). For instance, for the column "Good" in the "Test 1 Outcomes" the value 0.525 was obtained using the calculation: Utility(Buy the car, Car = Good) times Probability(Test 1 = Good / Car = Good) times Probability(Car=Good) plus Utility(Buy the car, Car = Bad) times Probability(Test 1 = Good / Car = Bad) times Probability(Car=Bad), or in numerical values $1 * 0.7 * 0.75 + 0 * 0.3 * 0.25 = 0.525$

Once all the entries in the top part of the table showing Test 1 outcomes were computed, find the maximum in each column (the max part of the formula), and add them up (the first summation of the formula). This will give the expected utility of the test. In this case the expected utility of Test 1 is 0.75. Repeating the process for Test 2 will give an expected utility of 0.9. So, we should perform Test 2 and if the car passes the test we should buy it.

Table B.9. Summary of the computation of the expected utility of Test 1.

Decisions	*Car's Condition*		*Test 1 Outcomes*	
	Good	Bad	Good	Bad
d_1= Buy the car	1	0	0.525	0.175
d_2= Do not buy the car	0	1	0.075	0.225
Probability(Car =)	0.7	0.3		
P(Test 1=Good/Car=)	0.75	0.25		
P(Test 1=Bad/Car=)	0.25	0.75		

In this example it was presented how to select an information based on the information reliability (the value of an information) and without considering the information cost. Utility theory can also include costs and personal preferences in the decision model. The reader interested in these more advanced topics should refer to Lindley ([44]).

BIBLIOGRAPHY

[1] Abramson, B., Brown, J., Edwards, W., Murphy, A., and Winkler, R.L. Hailfinder: a bayesian system for forecasting severe weather. *International Journal of Forecasting 12* (1996), 57–71.

[2] Andersen, S.K., Olesen, K.G., Jensen, F.V., and F., Jensen. Hugin - a shell for building bayesian belief universes for expert systems. In *Proceedings of the 11th International Congress on Uncertain Artificial Intelligence* (1989), pp. 1080–1085.

[3] Andreassen, S., Jensen, F.V., Andersen, S.K., Falck, B., U., Kjaerulff, M., Woldbye, A., Sorensen, A., Rosenfalck, and F., Jensen. Munin-an expert emg assistant. In *Computer-Aided electromyography and expert systems*, J.E. Desmedt, Ed. Amsterdam:Elsevier Science, 1989.

[4] Bacchus, F., and Grove, A.J. Utility independence in a qualitative decision theory. In *Proceedings of the Fifth International Conference on the Principles of Knowledge Representation and Reasoning* (1996), pp. 542–552.

[5] Bentham, J. *Principles of Morals and Legislation.* Oxford: Oxford University Press, 1789.

[6] Berler, A., and Shimony, S.E. Bayes networks for sensor fusion in occupancy grids. In *Proceedings of Uncertainty in Artificial Intelligence* (1997).

[7] Brooks, R.A. Symbolic reasoning among 3-d models and 2-d images. *Artificial Intelligence 17* (1981), 285–348.

[8] Brown, C.M., Marengoni, M., and Kardaras, G. Bayes nets for selective perception and data fusion. In *Proceedings of the SPIE on Image and Information Systems: Applications and Opportunities* (1994).

[9] Brunn, A., and Weidner, U. Discriminating buildings and vegetation areas within digital surface models. Tech. Rep. TR 97/2, Institute of Photogrammetry - University of Bonn, 1997.

[10] Buxton, H., and Gong, S. Advanced visual surveillance using bayesian networks. In *Proceedings of the IEEE Workshop on Context Based Vision, Cambridge, MA* (1995).

[11] B.W., Lindgren. *Elements of Decision Theory.* The Macmillan Company, New York, 1971.

[12] Castillo, E., Gutierrez, J.M., and Hadi, A.S. *Expert Systems and Probabilistic Network Models.* Springer, 1997.

[13] Cheng, J., Bell, D., and Liu, W. An algorithm for bayesian belief network construction from data. In *Proceedings of the AI & STAT, Ft Lauderdale, Florida* (1997), pp. 83–90.

[14] Cheng, J., Bell, D., and Liu, W. Learning bayesian networks from data: An efficient approach based on information theory. Tech. Rep. -, Department of Computer Science - University of Alberta, 1998.

[15] Collins, R., Cheng, Y., Jaynes, C., Stolle, F., Wang, X., Hanson, A., and Riseman, E. Site model acquisition and extension from aerial images. In *Proceedings of the Interantional Conference on Computer Vision* (1995), pp. 888–893.

[16] Collins, R., Jaynes, C., Y. Cheng, X. Wang, Stolle, F., Hanson, A., and Riseman, E. The ascender system: Automated site modelling from multiple aerial images. *Computer Vision and Image Understanding - Special Issue on Building Detection and Reconstruction from Aerial Images* (1998), to appear.

[17] Cooper, G.F. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial Intelligence 42* (1990), 393–405.

[18] Cooper, G.F., and Herskovits, E. A bayesian method for constructing bayesian belief networks from databases. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence* (1991), pp. 86–94.

[19] Crevier, D., and Lepage, R. Knowledge-based image understanding systems: a survey. *Computer Vision and Image Understanding 67(2)* (1997), 161–185.

[20] Debreu, G. *Theory of Value: An axiomatic analysis of economic equilibrium.* New York: Wiley, 1959.

[21] Draper, B., Collins, R., Broglio, J., Hanson, A., and Riseman, E. The schema system. *International Journal of Computer Vision 2* (1989), 209–250.

[22] Draper, B.A., Hanson, A.R., and Riseman, E.M. Knowledge-directed vision: control, learning, and integration. *Proceedings of the IEEE 84(11)* (1996), 1625–1637.

[23] Friedman, N., and Goldszmidt, M. Learning bayesian networks with local structures. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence* (1996), pp. 252–262.

[24] Haddawy, P., and Hanks, S. Representations for decision-theoretic planning: Utility functions for deadline goals. In *Proceedings of the Third International Conference on the Principles of Knowledge Representation and Reasoning* (1992), pp. 71–82.

[25] Hanson, A.R., and Riseman, E.M. Visions: A computer system for interpreting scenes. In *Computer Vision Systems*, A.R. Hanson and E.M. Riseman, Eds. Academic Press, 1978.

[26] Haralick, R.M., and Shapiro, L.G. *Computer and Robot Vision.* Addison-Wesley, 1993.

[27] Heckerman, D. A tutorial on learning with bayesian networks. Tech. Rep. MSR-TR-95-06, Microsoft Research, March 1995.

[28] Heckerman, D., Breese, J., and K., Rommelse. Decision-theoretic troubleshooting. *Communications of the ACM 38(3)* (1995), 49–56.

[29] Heckerman, D., E., Horvitz, and B., Nathwani. Towards normative experts sytems: Part i. the pathfinder project. *Methods of Information in Medicine 31* (1992), 90–105.

[30] Herman, M., and Kanade, T. Incremental reconstruction of 3-d scenes from multiple complex images. *Artificial Intelligence 30* (1986), 289–341.

[31] Horvitz, E.J., and B., Barry. Display of information for time-critical decision making. In *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence* (1995), pp. 296–305.

[32] Howard, R.A., and Matheson, J.E. Influence diagrams. In *Readings on the principles and applications of decision analysis*, R.A. Howard and J.E. Matheson, Eds. Strategic Decision Group, Menlo Park, California, 1984, pp. 721–762.

[33] Jaynes, C., and et.al. Three-dimensional grouping and information fusion for site modeling from aerial images. *DARPA Image Understanding Workshop* (1996), 479–490.

[34] Jaynes, C., Hanson, A., Riseman, E., and Schultz, H. Building reconstruction from optical and range images. In *Proceedings of the IEEE Computer Vision and Pattern Recognition, San Juan, Puerto Rico* (1997), pp. 380–386.

[35] Jaynes, C., Marengoni, M., Hanson, A., and Riseman, E. 3d model acquisition using a bayesian controller. In *Proceedings of the International Symposium on Engineering of Intelligent Systems, Tenerife, Spain* (1998).

[36] Jaynes, C., Marengoni, M., Hanson, A., and Riseman, E. Intelligent control for automatic model acquisition from aerial images. In *Proceedings of the IASTED International Conference on Intelligent Systems and Control, Halifax, Canada* (1998), pp. 30–35.

[37] Jaynes, C., Stolle, F., and Collins, R. Task driven perceptual organization for extraction of rooftop polygons. *IEEE Workshop on Applications of Computer Vision* (1994), 152–159.

[38] Jensen, F.V. *An introduction to Bayesian networks.* Springer Verlag New York, 1996.

[39] Jiang-Ming, L., Christensen, H.I., and Jensen, F.V. Qualitative recognition using bayesian networks. Tech. rep., Aalborg University - Institute of Eletronic Systems, 1993.

[40] J.O., Berger. *Statistical Decision Theory, Foundations, Concepts, and Methods.* Springer Verlag, 1980.

[41] Kahneman, D., and Tversky, A. Prospect theory: an analysis of decision under risk. *Econometrica 47* (1979), 263–291.

[42] Krebs, B., Burkhardt, M., and Korn, B. A task driven 3d object recognition system using bayesian networks. In *Proceedings of the International Conference on Computer Vision, Bombay, India* (1998), pp. –.

[43] Kumar, V.P., and Desai, U.B. Image interpretation using bayesian networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence 18(1)* (1996), 74–77.

[44] Lindley, D.V. *Making Decisions: Second Edition.* John Wiley and Sons, 1985.

[45] Machina, M.J. Dynamic consistency and non-expected utility models of choice under uncertainty. *Journal of Economic Literature 27(4)* (1989), 1622–1668.

[46] Mann, W.B., and Binford, T.O. An example of 3-d interpretation of images using bayesian networks. *DARPA Image Understanding Workshop* (1992), 793–801.

[47] Marengoni, M., Hanson, A., Zilberstein, S., and Riseman, E. Control in a 3d reconstruction system using selective perception. In *Proceedings of the International Conference in Computer Vision, Corfu, Greece* (1999), p. To appear.

[48] Marengoni, M., Jaynes, C., Hanson, A., and Riseman, E. Ascender ii, a visual framework for 3d reconstruction. In *Proceedings of the International Conference on Vision Systems, Las Palmas, Spain* (1999).

[49] McKeown, D.M., Jr., A.H. Wilson, and McDermott, J. Rule-based interpretation of aerial imagery. *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-7* (1985), 570–585.

[50] Pearl, J. *Probabilistic Reasoning in Intelligent System: Networks of Plausible Inference.* Morgan Kaufmann, 1988.

[51] Pinz, A., and Prantl, M. Active fusion for remote sensing image understanding. In *Image and Signal Processing for Remote Sensing II*, J. Desachy, Ed. SPIE Proceedings, vol 2579, 1995.

[52] Rasmussen, L.K. Bayesian networks for blood typing and parentage verification of cattle. Tech. Rep. Dina research report 38, Aalborg University - Department of Mathematics and Computer Science, 1995.

[53] Rimey, R., and Brown, C. Task-oriented vision with multiple bayes nets. In *Active Vision*, Blake A. and Yuille A., Eds. The MIT Press, 1992.

[54] Russell, S, and Norvig, P. *Artificial Intelligence: a modern approach.* Prentice Hall, 1995.

[55] Sarkar, S., and Boyer, K.L. Using perceptual inference networks to manage vision processes. *Computer Vision and Image Understanding 62(1)* (1995), 27–46.

[56] Schultz, H. Terrain reconstruction from widely spaced image. In *Integrating Photogrammetric Techniques with Scene Analysis and Machine Vision II, SPIE Proceedings, Vol 2486* (1995), pp. 113–123.

[57] Sucar, L.E., and Gillies, D.F. Probabilistic reasoning in high-level vision. *Image and Vision Computing 12(1)* (1994).

[58] von Neumann, J., and Morgenstern, O. *Theory of games and economic behavior.* Princeton University Press, 3rd edition, 1953.

[59] Wang, C.H., and Srihari, S.N. A framework for object recognition in a visually complex environment and its application to locating address blocks on mail pieces. *International Journal of Computer Vision 2* (1988), 125–151.

[60] Zilberstein, S. Using anytime algorithms in intelligent systems. *AI Magazine 17(3)* (1996), 73–83.