GOAL-DIRECTED CONTROL FOR COMPUTER VISION

Charles A. Kohl

COINS Technical Report 88-22

April 1988

Goal–Directed Control for Computer Vision

A Dissertation Presented

by

Charles Ainsworth Kohl

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

Doctor of Philosophy

February 1988

Department of Computer and Information Science

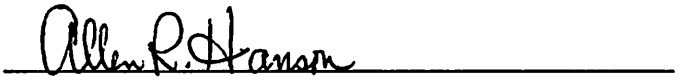
Goal–Directed Control for Computer Vision

A Dissertation Presented

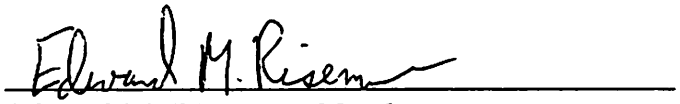
by

Charles Ainsworth Kohl

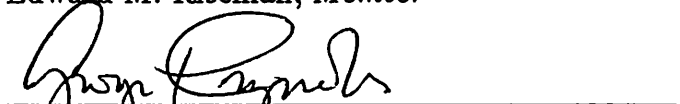
Approved as to style and content by:



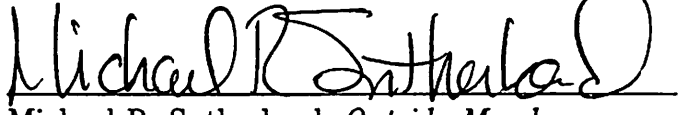
Allen R. Hanson, *Chairman*



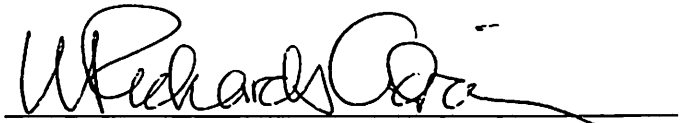
Edward M. Riseman, *Member*



George D. Reynolds, *Member*



Michael R. Sutherland, *Outside Member*



W. Richards Adrion, *Department Chairman*
Department of Computer and Information Science

© Copyright by Charles Ainsworth Kohl 1988

All Rights Reserved

This research was supported in part by the National Science Foundation Grant DCR-8318776 and the Air Force Office of Scientific Research Grant AFOSR-86-0021.

Dedicated to

loved ones of the past,

Big Charlie and Sissy,

and loved ones of the future,

Lauren and Kaitlin.

ABSTRACT

Goal–Directed Control for Computer Vision

February 1988

Charles Ainsworth Kohl, A.B. Princeton University

M.S. University of Massachusetts

Ph.D. University of Massachusetts

Directed by Professor Allen R. Hanson

Image interpretation is a complex process through which a numeric array, representing a digitized visual scene, can be analyzed to provide a semantic description of the scene content. One subgoal of the interpretation process is image segmentation, the low-level process by which the digitized image is abstracted into a set of primitive elements that may be used as the basis for the construction of an abstract symbolic model of the original scene. A commonly accepted view is that image segmentation is simply the first stage of the interpretation process. We take the view, however, that segmentation is a process which does not exist in isolation, but rather as an integral part of the overall image interpretation process.

This dissertation presents GOLDIE, an intermediate-level, goal-directed system that has been developed within the VISIONS system to accomplish the integration of the the high and low levels of the image understanding process. GOLDIE is a system designed to operate in either a data–driven or interpretation–driven manner. In the data–driven mode, GOLDIE functions as a segmentation system that is able to choose between a variety of algorithms and image features based on the characteristics of the image data. In the interpretation–driven mode, the system provides the the mechanism by which high-level requests for data (goals) can activate a set of appropriate low or intermediate-level processes that may be capable of producing the desired data. Through the use of the information expressed on the goal, GOLDIE is able to select the most appropriate image features, algorithms, sensitivity settings, rules, etc., that tune these processes to the production of the desired data.

Contents

Abstract v

List of Tables xi

List of Figures xii

Chapter

1. Goal–Directed Control for Computer Vision 1

1.1 Introduction 1

1.2 VISIONS: Moving from Raw Data to Abstract Structure to Hypothesis 4

1.2.1 The Low Level 6

1.2.2 The Intermediate Level 7

1.2.3 The High Level 8

1.3 The Role of Low Level Processing 9

1.3.1 Image Segmentation 10

1.3.2 The Difficulty of Image Segmentation 11

1.4 Design of GOLDIE 16

1.4.1 Segmentation Processing 16

1.4.2 Data Structures 18

1.4.3 Knowledge Representation and Control 19

1.5 The Structure of GOLDIE 20

1.5.1 Image Data 22

1.5.2 Process Controller 24

1.5.3 Intermediate Long Term Memory (ILTM) 24

1.5.4 Intermediate Short Term Memory (ISTM) 24

1.5.5 Goal Blackboard 25

1.5.6 Intermediate–level Schema Instantiations 25

1.6 Environment 25

1.7 Experimental Protocol 26

1.8 Contributions 26

2. Review of Relevant Literature **28**
2.1 Introduction 28
2.2 Image Representations 29
2.2.1 Color Space 30
2.2.2 Texture 31
2.3 An Overview of Segmentation Techniques 33
2.3.1 Region–Based Segmentation Techniques 33
2.3.1.1 Region Growing Techniques 34
2.3.1.2 Split–and–Merge 36
2.3.1.3 Measurement Space Clustering 38
2.3.1.4 Characteristics of Region–Based Techniques . . . 43
2.3.2 Edge–Based Segmentation Techniques 44
2.3.2.1 Edge–Based Region Segmentation 45
2.3.2.2 Edge Detection 45
2.3.2.3 Segmentation from Edges 48
2.3.2.4 Characteristics of Edge–Based Techniques 50
2.3.3 Techniques Using a Combined Region/Edge Representation 51
2.4 Knowledge–Based Control of Low and Intermediate–Level Processing 53
2.4.1 Data–Directed Semantic Grouping Systems 54
2.4.2 Top–Down Goal–Directed Segmentation 59
2.4.3 Integrated Top–Down and Bottom–Up Systems 61
2.5 Conclusion 64

3. Low–Level and Intermediate–Level Processes **66**
3.1 Image Features 66
3.1.1 Color Image Features 71
3.1.2 Texture Image Features 79
3.1.3 Semantic Image Features 87
3.2 Image Enhancement 89
3.3 Region–Based Segmentation 92
3.3.1 Thresholding 93
3.3.2 Zero–Crossings 96
3.3.3 One–Dimensional Histogram Clustering 99
3.3.3.1 One–Dimensional Global Histogram Clustering . . 100
3.3.3.2 One–Dimensional Localized Histogram Clustering 102

3.3.4 Two–Dimensional Histogram Clustering 105
3.3.5 Segmentation Postprocessing 110
3.3.5.1 Small Region Suppression 111
3.3.5.2 Plurality Postprocessing 112
3.4 Region Merging 114
3.4.1 Rules For Region Merging 114
3.4.2 Region Merge Processing 121
3.5 Region Evaluation 122
3.5.1 Data–Oriented Region Evaluation 122
3.5.2 Semantic Region Evaluation 130
3.6 Lines from Zero–Crossings 132
3.6.1 Edges from Zero–Crossings 132
3.6.2 Collinear Line Grouping 134
3.7 Gradient–Orientation Based Line Extraction 138
3.8 Line Segmentation 140
3.8.1 Direct Line Insertion 142
3.8.2 Line–Based Segmentation 147
3.9 Low–Level Process Controller 149
3.10 Conclusion 152

4. Intermediate Level Data Structures of GOLDIE 153
4.1 Introduction 153
4.2 Data structure Requirements 154
4.3 Semantic Network Representation 155
4.3.1 Utility of the Network Representation 156
4.3.2 Implementation of the Semantic Network in GOLDIE . . . 158
4.4 Intermediate Symbolic Representation 159
4.5 Intermediate–Level Representation of GOLDIE 161
4.6 Dynamic data structures of ISTM 163
4.6.1 Image Data Structures 163
4.6.2 Process Descriptions 167
4.6.3 Hypotheses 169
4.6.4 Region Tokens 169
4.6.5 Line Tokens 173
4.7 Knowledge Structures of ILTM 175

4.8 Control State Representation 177
4.9 Use of the Intermediate–Level Structures in Interpretation 179
4.10 Discussion 180

5. Schemas and Control 181
5.1 Requirements for Representation of Knowledge and Control in GOLDIE 181
5.2 Schema Control Mechanisms 182
5.2.1 Schema Structures 183
5.2.2 Schema Operation 187
5.2.3 Evaluation Constraints 188
5.3 Example of Schema Control 189
5.4 Intermediate–Level Schemas of GOLDIE 191
5.4.1 The IDDS Schema 194
5.4.2 The Region Schemas 204
5.4.3 The Line Schemas 219
5.5 Discussion 222
5.5.1 Knowledge Representation 223
5.5.2 Control 225
5.5.2.1 Production–Rule Systems 225
5.5.2.2 Means–End Analysis 226
5.5.2.3 Frame–Based Systems 227
5.5.3 Conclusion 227

6. Experimental Demonstration of GOLDIE 229
6.1 Introduction 229
6.2 Data–Directed Processing 230
6.2.1 Initial Segmentations 230
6.2.2 Intermediate–Level Control in the IDDS Schema 271
6.2.3 Initial Segmentations Produced Under *a priori* Constraints 291
6.3 Expectation–Driven Processing 297
6.3.1 Expectation–Driven Processing for Object Extraction . . . 300
6.3.2 Extraction of Shape Constrained Tokens 310
6.3.3 Extraction of Obscure Image Tokens 317
6.4 Conclusion 322

7. Contributions and Future Research 328
7.1 Contributions 328
7.1.1 Knowledge of the Image Domain 328
7.1.2 Intermediate–Level Data Representation 329
7.1.3 Control 330
7.2 Future Research 332
7.3 Conclusions 332

Bibliography 333

List of Tables

3.1 Image Features of the GOLDIE System 68
3.2 Color Image Features 72
3.3 Texture Image Features 80
3.4 Semantic Image Features 88
3.5 Region Merge Evaluation Rules 119
3.6 Smooth Region Merge Evaluation Rule Set 120
3.7 Gradient Region Merge Evaluation Rule Set 120
3.8 Texture Region Merge Evaluation Rule Set 121
3.9 Region Evaluation Rules 123
3.10 Smooth Region Resegmentation Evaluation Rule Set 124
3.11 Gradient Region Resegmentation Evaluation Rule Set 124
3.12 Texture Region Resegmentation Evaluation Rule Set 125
3.13 Smooth Region Merge–Potential Evaluation Rule Set 127
3.14 Gradient Region Merge–Potential Evaluation Rule Set 128
3.15 Texture Region Merge–Potential Evaluation Rule Set 128
4.1 ISR Region Lexicon 172
4.2 ISR Line Lexicon 174
5.1 Intermediate–Level Schema Descriptions 193
5.2 Constraints Known to the IDDS Schema 195
5.3 Constraints Known to the Region–Segmentation Schema 206
5.4 Constraints Known to the Region–Feature Schema 212
5.5 Constraints Known to the Region–Algorithm Schema 213
5.6 Constraints Known to the Region–Evaluation Schema 215
5.7 Constraints Known to the Region–Merge–Schema 217
5.8 Constraints Known to the Line–Extraction Schema 219
5.9 Constraints Known to the Line–Segmentation Schema 221
5.10 Constraints Known to the Collinear–Line–Merge Schema 222
6.1 Segmentation Process Specifications from IDDS Schema 272

List of Figures

1.1 The VISIONS Image Understanding System 5
1.2 Intensity Image of Outdoor Scene 13
1.3 Zero–Crossing Segmentation 14
1.4 Histogram Clustering Segmentation 15
1.5 Position of GOLDIE in the VISIONS Hierarchy 16
1.6 Overview of the GOLDIE System 21
1.7 The Processing Cone 23
3.1 Feature Node 68
3.2 Typical Manufacturing Scene 69
3.3 Tactile Image of a Key 70
3.4 Hue Image Feature 73
3.5 Saturation Image Feature 74
3.6 Frequency Distribution of Normred Before Scaling 75
3.7 Frequency Distribution of Normred After Scaling 76
3.8 Itwor Image Feature 77
3.9 Itwog Image Feature 78
3.10 Itwob Image Feature 79
3.11 Q Image Feature 81
3.12 Mean Image Feature 82
3.13 Variance Image Feature 83
3.14 Deviation Image Feature 84
3.15 Extremadensity Image Feature 85
3.16 Energy Image Feature 86
3.17 Entropy Image Feature 87
3.18 Objfeat-grass-tree Image Feature 89
3.19 Objfeat-bush-tree Image Feature 90
3.20 Threshold Segmentation - One Threshold 95
3.21 Threshold Segmentation - Two Thresholds 96
3.22 Threshold Segmentation - Three Thresholds 97
3.23 Zero–crossing Segmentation - Low Sensitivity 98

3.24 Zero–crossing Segmentation - Medium Sensitivity 99
3.25 Zero–crossing Segmentation - High Sensitivity 100
3.26 Global One–Dimensional Segmentation - Low Sensitivity 102
3.27 Global One–Dimensional Segmentation - Medium Sensitivity . . . 103
3.28 Global One–Dimensional Segmentation - High Sensitivity 104
3.29 Localized One–Dimensional Segmentation - Low Sensitivity 106
3.30 Localized One–Dimensional Segmentation - Medium Sensitivity . . 107
3.31 Localized One–Dimensional Segmentation - High Sensitivity 108
3.32 Two–Dimensional Histogram of Intensity vs. Itwob 109
3.33 Two–Dimensional Histogram Peaks - Low Sensitivity 110
3.34 Two–Dimensional Histogram Peaks - High Sensitivity 111
3.35 Global Two–Dimensional Segmentation - Low Sensitivity 112
3.36 Global Two–Dimensional Segmentation - Medium Sensitivity . . . 113
3.37 Global Two–Dimensional Segmentation - High Sensitivity 114
3.38 Two–Dimensional Histogram of Red vs. Blue 115
3.39 Small Region Suppression 116
3.40 Plurality Postprocessing 117
3.41 Resegmentation Hypotheses with Smooth Constraint 126
3.42 Resegmentation Hypotheses with Texture Constraint 127
3.43 Resegmentation Hypotheses with Gradient Constraint 128
3.44 Merge–Potential Hypotheses with Smooth Constraint 129
3.45 Merge–Potential Hypotheses with Texture Constraint 130
3.46 Merge–Potential Hypotheses with Gradient Constraint 131
3.47 3x3 Laplacian Mask 133
3.48 Image Data 134
3.49 Zero–Crossing Contours 135
3.50 Edges Produced by Zero–Crossing Based Algorithm 136
3.51 Edges from Figure 3.50 That Have High Contrast 136
3.52 Collinear Line Grouping 137
3.53 2x2 Masks for Gradient–Orientation Based Lines 138
3.54 Lines Produced by Gradient Orientation Algorithm 140
3.55 High Contrast Lines Produced by Gradient Orientation Algorithm 141
3.56 Lines Produced by Gradient Orientation Algorithm 141
3.57 Slfeat Image Feature 142
3.58 Initial Region Segmentation for Direct Line Insertion 143

3.59 The Set of Lines for Direct Line Insertion 144
3.60 Intersecting Regions for Direct Line Insertion 145
3.61 Region Segmentation Resulting from Direct Line Insertion Process 146
3.62 Region Segmentation for Line–Based Segmentation 148
3.63 Line for Line–Based Segmentation 149
3.64 Temporary Regions for Line–Based Segmentation 150
3.65 New Regions Produced by Line–Based Segmentation 151
4.1 Overview of the GOLDIE System 162
4.2 Image Node Structure 164
4.3 Hierarchical Image Node Structure 166
4.4 Region Segmentation Node Representation 168
4.5 Line–Extraction Node Representation 168
4.6 Goal Node Representation 178
5.1 Contract Structure 186
5.2 Region for Resegmentation 190
5.3 Goal Node 190
5.4 Schemas of the GOLDIE System 192
5.5 Top–level Segmentation Produced by IDDS Schema 198
5.6 Long High Contrast Lines 199
5.7 Insertion of Long High Contrast Lines 200
5.8 Regions with Strong Resegmentation Hypotheses 201
5.9 Tree/Bush Region That Has Strong Resegmentation Hypothesis . 202
5.10 Resegmentation of Tree/Bush Region 203
5.11 Final Result from Resegmentation of Tree/Bush Region 204
5.12 IDDS Schema Segmentation Prior to Final Merge 205
5.13 Segmentation from IDDS Schema 207
5.14 Image with Overlaid Segmentation from IDDS Schema 208
5.15 Region Segmentation that was Rejected by Region Segmentation Schema 209
5.16 Region Segmentation that was Rejected by Region Segmentation Schema 210
5.17 Pseudocode for Region–Segmentation Schema 211
6.1 Outdoor House Scene (House 15) 233
6.2 Oversegmentation of House 15 234
6.3 Region–Merge Segmentation of House 15 235

6.4 Partitioned One–Dimensional Segmentation of House 15 (Medium) 236
6.5 Localized One–Dimensional Segmentation of House 15 (Medium) . 237
6.6 Localized One–Dimensional Segmentation of House 15 (Very–Low) 238
6.7 Localized One–Dimensional Segmentation of House 15 (Low) . . . 239
6.8 Localized One–Dimensional Segmentation of House 15 (High) . . . 240
6.9 Localized One–Dimensional Segmentation of House 15 (Very–High) 241
6.10 GOLDIE IDDS Schema Segmentation of House 15 244
6.11 House 15 Image with IDDS Segmentation 245
6.12 Outdoor House Scene (House 1) 246
6.13 Region–Merge Segmentation of House 1 247
6.14 Localized One–Dimensional Segmentation of House 1 (High) 248
6.15 Localized One–Dimensional Segmentation of House 1 (Very–High) 249
6.16 GOLDIE IDDS Schema Segmentation of House 1 250
6.17 Outdoor Road Scene (Road 16) 251
6.18 Region–Merge Segmentation of Road 16 252
6.19 Localized One–Dimensional Segmentation of Road 16 (Very–High) 253
6.20 GOLDIE IDDS Schema Segmentation of Road 16 254
6.21 Indoor Scene (Room 50) 255
6.22 Region–Merge Segmentation of Room 50 256
6.23 Localized One–Dimensional Segmentation of Room 50 (High) . . . 257
6.24 GOLDIE IDDS Schema Segmentation of Room 50 258
6.25 Outdoor Road Scene (Road 1) 259
6.26 Region–Merge Segmentation of Road 1 260
6.27 Localized One–Dimensional Segmentation of Road 1 (Very–High) . 261
6.28 GOLDIE IDDS Schema Segmentation of Road 1 262
6.29 Outdoor Road Scene (Road 25) 263
6.30 Region–Merge Segmentation of Road 25 264
6.31 Localized One–Dimensional Segmentation of Road 25 (Very–High) 265
6.32 GOLDIE IDDS Schema Segmentation of Road 25 266
6.33 Outdoor House Scene (House 7) 267
6.34 Region–Merge Segmentation of House 7 268
6.35 Localized One–Dimensional Segmentation of House 7 (High) 269
6.36 GOLDIE IDDS Schema Segmentation of House 7 270
6.37 Intensity-1D-Plurality Segmentation 273
6.38 Red-1D-Plurality Segmentation 274

6.39 Itwob-1D-Plurality Segmentation 275
6.40 Intensity/Hue-2D-Plurality Segmentation 276
6.41 Slfeat-1D-Plurality Segmentation 277
6.42 Blue-1D-Plurality Segmentation 278
6.43 Green-1D-Plurality Segmentation 279
6.44 Normgreen-1D-Plurality Segmentation 280
6.45 Lines for House 15 Image 282
6.46 Long, High–Contrast Lines for House 15 Image 283
6.47 Long Line Insertion for House 15 Image 284
6.48 Resegmentation Hypothesis Regions 286
6.49 Goal for the Resegmentation of Grass Region 286
6.50 Resegmentation of Grass Region 288
6.51 Goal for the Resegmentation of the Bush/Wall Region 288
6.52 Resegmentation of Bush/Wall Region 289
6.53 Goal for the Resegmentation of the Bush Region 290
6.54 Resegmentation of Bush Region 292
6.55 Segmentation Prior to Final Merging 294
6.56 Results of Region–Merge Schema with Default Constraint 295
6.57 Results of Region–Merge Schema with Texture Constraint 296
6.58 Results of IDDS Schema 297
6.59 IDDS Segmentation With Smooth Constraint 298
6.60 IDDS Segmentation With Texture Constraint 299
6.61 IDDS Segmentation With Gradient Constraint 300
6.62 Goal Specification for Road Segmentations 300
6.63 Constrained IDDS Segmentation for Road 1 301
6.64 Constrained IDDS Segmentation for Road 16 302
6.65 Constrained IDDS Segmentation for Road 25 303
6.66 Barn in Segmentation of Road 16 304
6.67 Barn in Resegmentation of Road 16 305
6.68 Image Data for Region Containing Tree and Sky 306
6.69 Tree/Sky Region for Resegmentation 307
6.70 Tree and Sky Regions 308
6.71 Bush/Grass Region 309
6.72 Resegmentation of Bush/Grass Region 310
6.73 Hypothesized Horizon Line 312

6.74 Line–Segmentation of Grass Region 313
6.75 IDDS Segmentation With Low–Resolution Constraint 314
6.76 Resegmentation of House Region 315
6.77 Resegmentation of Foliage Region 316
6.78 Resegmentation of House 15 Image 317
6.79 Goal for Resegmentation of Shutters 317
6.80 IDDS Segmentation of Portion of House 1 319
6.81 Rectangular Shutter Regions 320
6.82 Telephone Pole 321
6.83 Line Tokens for Road 16 Image 322
6.84 Parallel Lines Near Hypothesized Telephone Pole 323
6.85 Additional Lines for Road 16 Image 324
6.86 New Set of Parallel Lines Near Hypothesized Telephone Pole . . . 325
6.87 Region Token for Telephone Pole 326
6.88 Goal for High–Resolution Segmentation of Sky Region 326
6.89 Oversegmentation of Sky Region 327
6.90 Telephone Wire Regions 328

Chapter 1

Goal–Directed Control for Computer Vision

1.1 Introduction

Image interpretation is a complex process through which a numeric array, representing a digitized visual scene, can be analyzed to provide a semantic description of the scene content. One subgoal of the interpretation process is **image segmentation**, the low–level process by which the digitized image is abstracted into a set of primitive elements that may be used by interpretation processes as the basis for the construction of an abstract symbolic model of the original scene. According to many commonly accepted views of interpretation, image segmentation is simply the first stage of the interpretation process. We take the view, however, that segmentation is a process which does not exist in isolation, but rather is an integral part of the overall image interpretation process.

This concept of the interdependence between the segmentation and interpretation processes has developed slowly over the years as researchers have become aware of the difficulty of each of the phases of processing. Researchers such as Tenenbaum and Barrow [183], Marr [127], Brooks and Binford [30], Thompson [184], Reynolds [164], Hwang [105] and Hanson and Riseman [166] have all stressed the need for interaction between high and low level processes. For example, Marr states:

> What was wrong with the idea of segmentation? The most obvious flaw seemed to be that "objects" and "desirable regions" were almost never visually primitive constructions, and hence could not be recovered from the primal sketch or other similar early representations

without additional specialized knowledge.([127, page 272])

To date, however, little has been done to address this problem, and thus there are no commonly accepted protocols for the interaction. If we are to adequately deal with interaction between high and low–level processes, there are four separate issues which must be resolved. First, there must be a common representation for data; intercommunication between processes operating at different levels of abstraction becomes difficult or impossible if data objects and hypotheses are not consistent across the (somewhat artificial) boundaries between the levels. Second, there is the necessity for a single unified control mechanism that can coordinate and schedule the activities of high and low–level processing tasks. Third, there must be a mechanism that can be used to evaluate the quality of the data that is produced by these processing tasks. Finally, there is the requirement for an explicit representation of the knowledge necessary for the effective application of low–level processes (i.e. the selection of the appropriate algorithms, parameter settings, and image features). For example, this representation must provide the ability to encode the knowledge that indicates that certain image features and algorithms are generally useful when segmenting textured image areas, while other features and algorithms are more useful when segmenting smooth image areas.

This dissertation will present a system that has been developed within the VISIONS Image Understanding System [82,83,113] to satisfy these three requirements and thereby accomplish the integration of the the high and low levels of the image understanding process. The system is designed to provide the mechanisms by which high–level requests for data may activate a set of appropriate low–level processes which may be capable of producing the desired data. The specification of the data request is in the form of a *goal*: a data structure that defines the nature of the image abstractions desired by the requesting process. *Goal constraints*, stored as attributes within the goal data structure, express additional information which may be useful in the specification of the most appropriate low–level

processes. These goal constraints express the desired characteristics of the data to be produced, and may be represented in either semantic or image–based terms. The semantic constraints are defined in terms of semantic labels (e.g. *"segment a specific portion of the image to separate Tree and Sky"*), while the image–based constraints are expressed in terms of measurable image features (e.g. *"produce regions which exhibit homogeneous texture measures"*). Through the use of the information expressed in these constraints, the system is able to select the most appropriate image features, algorithms, sensitivity settings, rules, etc., for the specification of the low–level task.

The interface between the high and low levels of the interpretation process in the VISIONS system is called the *intermediate* level; data at this level is represented by *tokens*. Tokens are symbolic entities that represent any abstraction of the raw image data which is in spatial registration with the actual data and/or groups of these abstractions which have been formed by the application of grouping operations [15,163,192]. These intermediate–level tokens may be used to represent any of the normal types of segmentation output, such a regions, lines, or surfaces, and in fact can include any event that is extractable from the sensory data (e.g. corners, textured areas, etc.). Attributes of the tokens encode measurable features of the token with respect to the image data (e.g. size, mean intensity, or location).

The GOLDIE (Goal Directed Intermediate–Level Exectutive) system has been developed as a mechanism to provide intermediate–level control of both low and intermediate–level processing, in effect making the data representation at the intermediate level a function not only of the low–level data–directed processes, but also of the expectations of the high–level interpretation system. The basic control structure of GOLDIE is the *schema*, a declarative specification of the *control* strategies which may be used by the system to satisfy a specific goal. Schemas provide a flexible and extensible control structure which is used within VISIONS to direct the high–level interpretation processes [51,83,152,197]. GOLDIE ex-

tends this concept of schema–directed control to the low and intermediate–level processes and provides a natural interface between processes at all levels of the abstraction hierarchy.

1.2 VISIONS: Moving from Raw Data to Abstract Structure to Hypothesis

The VISIONS Image Understanding System [80,113] (summarized in Figure 1.1) may be viewed as being organized into three distinct levels of abstraction: the low, intermediate, and high levels. These three levels provide a structure for the representation of data and knowledge as well as the control of the processes which make use of this data and knowledge.

The low level of the VISIONS system is concerned with image data; the knowledge at this level consists of the procedural representation of operators which can be used to extract information from the image array. The intermediate–level provides a symbolic representation of image data; in this case knowledge is used to control the *knowledge sources* which evaluate and manipulate the token data. High level knowledge is used to provide an interpretation of the intermediate–level tokens.

With respect to this hierarchy, control of processing may be viewed as top–down, bottom–up, or some combination of both. Here we define bottom–up processing as data–directed in the sense that the existence of raw data or image tokens allows the creation of tokens or hypotheses at a higher level of the interpretation hierarchy. Bottom–up processes are knowledge–free in the sense that they are applied uniformly across the image data without knowledge of the image contents. As an example, the existence of a set of adjacent region tokens which exhibit low contrast across adjacent boundaries permits a bottom–up grouping process to create a new region token. Top–down control, on the other hand, is characterized by hypothesis–driven processes in which image tokens or hypothe-

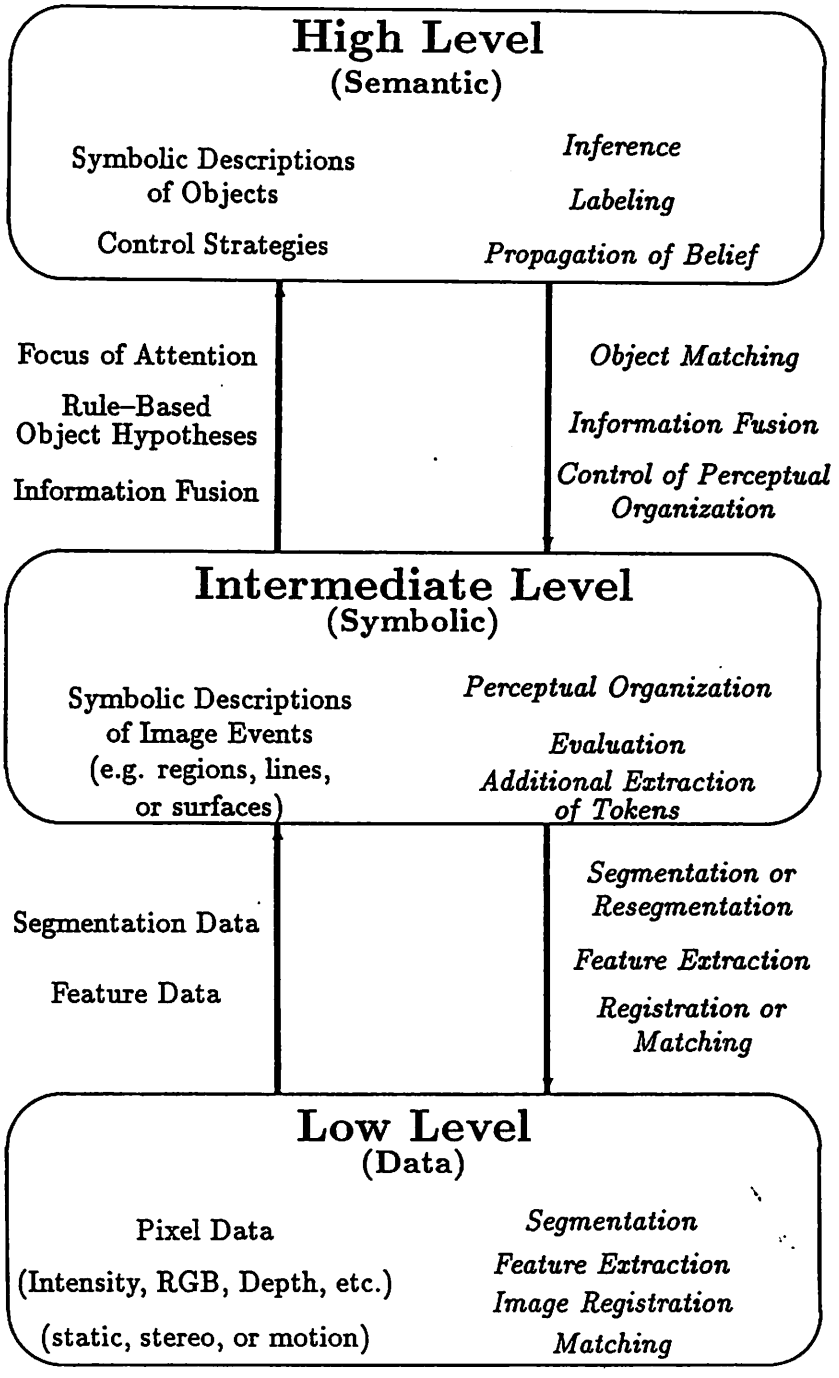


Figure 1.1: The VISIONS Image Understanding System

ses are created or verified at a lower level of abstraction in the interpretation hierarchy. An example of this form of control would be a situation in which an interpretation process directs the merge of adjacent region tokens which have similar object labels. In the following sections, each of these levels is discussed in more detail.

1.2.1 The Low Level

The low level of VISIONS (the bottom rectangle of Figure 1.1) is concerned with the abstraction of raw image events into symbolic tokens; this abstraction takes place through various data–directed processes such as region segmentation or line extraction. In order to achieve this overall objective, the various processes at this level are designed to serve a number of functions. Since these processes are all concerned with image data, one significant function of this level is to maintain a database for the raw images and the image feature data that has been computed from the raw data. Other forms of two–dimensional data which are in registration with the image data (e.g. depth data) are also stored within this database structure.

A second function of the low level is the control of *image processing* (i.e. the specification and control of those processes that manipulate image data to produce a new form of image data). Operators which provide the capability for image enhancement or image restoration allow the image data to be conditioned to remove noise or to compensate for distortions that arise during the imaging and digitization processes. Registration operators are used to produce a sequence of images which have been correctly aligned to compensate for sensor motion, and low–level motion and stereo operators can be used to construct displacement fields used in the calculation of depth arrays.

The third major function of this level is *segmentation processing*. Where the image processing tasks are concerned with the production of a new form of image data, segmentation tasks are concerned with the production of image abstrac-

tions (i.e. tokens). A variety of operators have been implemented at this level to perform such operations as region segmentation [82,114,115,137], line extraction [34], stereo matching [2,199], depth extraction from motion information [19,20], etc.. Since each of these processes may be run with a wide variety of parameter settings on a variety of image features, the VISIONS system embeds the processes in a LISP environment so that a user may experimentally develop low–level algorithms. A major advantage of this approach is that the environment within which these processes are run is compatible with the environment provided at the high level: thus it becomes possible to establish natural channels of communication between the various levels of the system.

1.2.2 The Intermediate Level

The intermediate level of processing (the middle rectangle of Figure 1.1) is concerned with the symbolic representation of significant events in the image data. These events are represented as tokens of the ISR (the Intermediate–level Symbolic Representation). The ISR functions as a database that stores both the spatial description of the tokens as well as the set of *token features* (values for tokens which may be calculated from the image data). Token features may either be calculated at the time of token creation or they may be computed on demand by the ISR. This representation provides an extremely flexible representation for image tokens and will be discussed in detail in Chapter 4.

Processes that operate at the intermediate level of VISIONS are primarily concerned with either focus–of–attention or the application of grouping operations to the tokens represented in the ISR. These processes may be initiated under high–level control, or may operate in a data–directed manner. When operating under high–level control, these intermediate level processes typically act in a very localized manner (e.g. merging two adjacent region tokens that some interpretation process believes represent the same semantic object). When operating in a data–directed fashion, the intermediate–level processes act in a more global

fashion, using prespecified criteria to control the creation or modification of a large number of the symbolic tokens. Data–directed grouping processes create new tokens that represent the union of sets of existing tokens, providing a form of *perceptual organization* at the intermediate level. An example of this type of grouping operation would be a region merge process that creates a new region token representing the union of two (or more) adjacent regions which exhibit similar feature characteristics and which have no high contrast line near their mutual boundary [18,140]. A different example would be a line grouping process that forms a new line representation from two existing lines which are spatially adjacent and collinear [163,192].

Focus–of–attention in the VISIONS system can be provided at both the intermediate and high levels of processing. At the intermediate level, this focus can be provided through rule–based operators which represent a set of constraints on the range of attribute values on the ISR tokens; e.g. constraints on the color of region tokens, or length of line tokens, or orientation of surface tokens. These constraints are used to rank order the potential object hypotheses for the tokens, effectively providing focus–of–attention for those high–level processes that are to create object hypotheses [15,166,196,197]. Other rule–based operators at the intermediate level may use constraints on relations between tokens that have been derived from different images to provide the focus–of–attention for stereo or motion processes [1,2,70].

1.2.3 The High Level

At the high level of processing (the top rectangle of Figure 1.1), the VISIONS system is concerned with the formation and verification of semantic hypotheses. Control of these processes is organized through LTM (Long Term Memory), a semantic network of schema nodes, each of which has both a declarative and a procedural component. The network is organized in terms of a compositional hierarchy of PART–OF relations and a subclass hierarchy of IS–A relations. The

interpretation processes construct a network in STM (Short Term Memory) that is composed of image–specific instances of portions of the LTM network. The schema knowledge structures provide the set of control strategies that are used by the system to instantiate the nodes of STM with respect to the tokens of the ISR [15,49,82,197,200]. Strategies such as constraint–based object matching, information fusion, and the control of intermediate–level grouping processes are used to create hypotheses about particular (sets of) tokens that form the partial interpretations which are represented by the STM nodes.

These partial interpretations are extended from "islands of reliability" as in the HEARSAY–II paradigm [56]. Inference mechanisms (e.g. the combination of uncertain evidence [162,164], the network propagation of belief [123,194,195,196], or the use of symbolic endorsements [41,51]) provide the mechanisms by which particular schema process may be invoked to extend the islands of reliability into a complete interpretation in STM.

1.3 The Role of Low Level Processing

Given this complex hierarchy of interacting processes involved in the image interpretation process, it becomes apparent that the behavior of the entire system is highly dependent upon the quality of the data produced by the low–level segmentation processes. If the set of tokens in the ISR does not provide an adequate description of the events in the image, the performance of the higher level processes that operate on this data can be seriously degraded.

In this section we will expand on the preceding discussion of low–level processing and demonstrate the need for intelligent intermediate–level control that served as the motivation for GOLDIE.

1.3.1 Image Segmentation

An image segmentation is defined as an abstraction of the array of raw image data. The vast amount of data contained in the original digitized image is abstracted into a set of symbolic data structures representing intermediate–level events such as regions, lines, or surfaces. These abstracted data elements are referred to as *tokens*, symbolic entities that have a specific mapping into image space. It is the set of tokens, rather than the raw pixel data, that is used by the higher–level interpretation processes as the basis for object identification.

In other words, segmentation processing is used to establish the domain for the interpretation processing. Except in the case of relatively simple classification problems (e.g. land use classification based on low resolution satellite imagery), the complexity of image data precludes the direct interpretation of the raw data. The abstract tokens produced through segmentation processing form a manageable data set over which interpretation can occur. Not only is the size of the initial data set reduced, but the characteristics of the intermediate–level tokens (e.g. size, shape, color, variance, etc.) are more closely related to the characteristics of the objects to be identified.

Since the segmentation process is viewed as an integral component of the overall image interpretation process, it follows that *the quality of a segmentation is dependent only upon its utility with respect to a set of interpretation goals and cannot be measured independently of the interpretation process.* These goals [69,174] can include such diverse purposes as object recognition, scene analysis, motion detection, or depth perception. Even within the domain of a specific goal, however, there may be no such thing as a perfect segmentation. Factors such as the precise placement of boundaries or the level of resolution for the region and line tokens make the segmentation process inexact (*cf.* Section 3.2). There may be a variety of possible segmentations for an image that satisfy the goals of the interpretation process. The "best" segmentation therefore becomes the one which serves both to maximize the satisfaction of interpretation goals while minimizing

the amount of segmentation processing.

With respect to specific interpretation goals, the "quality" of a segmentation may be measured in a strictly data–directed manner. An example would be a system for the analysis of Landsat data [205] in which the segmentation serves as the basis for statistical evaluation of image features associated with terrain features. In this domain, the segmentation process is used to constrain the pixel classification problem. Regions are statistically classified on the basis of the characteristics of the region, and all pixels contained in the region are given this classification. In such a case, the "best" segmentation is the one for which the classification process is able to assign the highest possible confidence to each region token of the segmentation. Thus, the evaluation of the segmentation may potentially be calculated from the confidence values themselves.

However, if the interpretation goal is to provide a complete semantic description of image content, the evaluation of the individual region tokens must be performed in a goal–directed manner relative to the needs of the interpretation process. In this case, localized semantic knowledge of the objects and/or scene represented in the image can be used to establish the intermediate–level evaluation criteria. Such an approach requires, at a minimum, that the intermediate–level system contain knowledge of the object domain of the image (the set of object classes to be identified), as well as knowledge of the image–level characteristics of tokens which represent these objects. It is only through the use of this type of knowledge that the evaluation of the segmentation data can become sensitive to the interpretation context.

1.3.2 The Difficulty of Image Segmentation

Low–level segmentation processing is in many ways more of an art than a science. Despite an enormous number of excellent techniques and algorithms (e.g. see [10,52,79,92,170]), no general methods have been found which perform adequately across a wide variety of images. Methods such as region-growing [85],

histogram cluster labeling [137], thresholding [114], zero-crossings [90,127], and edge and line detection [34,37] have all been found to be effective within restricted domains, but rarely demonstrate the robustness necessary for generalized image interpretation.

The failure of general segmentation techniques to provide "good" intermediate-level data can be traced to a variety of factors. In many cases the image data itself is complex because of the physical situation from which the image was derived and the complexity of the scene. Heavy textures, unconstrained lighting and variable view angle, surface reflectivities, and curvature all conspire to produce ambiguous image data. This results in undermerged or overmerged regions in the region segmentations and missing, fragmented, or incorrectly merged lines in the edge/line segmentations. The type of error often depends upon which image feature(s) is being used to produce the segmentation and whether the algorithm has a global or local view [137] of the image data during processing. The type of error produced quite often varies spatially within a single image, depending on the type of data in each locality of the image.

To take a specific example, the intensity surface of an image (Figure 1.2) may contain a great deal of high frequency information which typically will lead to significant *fragmentation* in the segmentation, such as shown in Figure 1.3. Here we see that the region segmentation, which was produced by a zero-crossing algorithm [127], contains a large number of small regions in the textured projection of the tree. While these regions do represent image areas of local homogeneity, in most cases they are of little semantic interest. If the overall interpretation goal is to identify the large objects in the scene, a segmentation such as that shown in Figure 1.4, which was produced through a one-dimensional histogram clustering algorithm (*cf.* Section 3.3.3), would be much more useful for the identification of the trees in the image.

In the case of the smoothly varying or flat surfaces of this image, however, the opposite is true. Here, the slow gradients of the intensity surface result in *miss-*

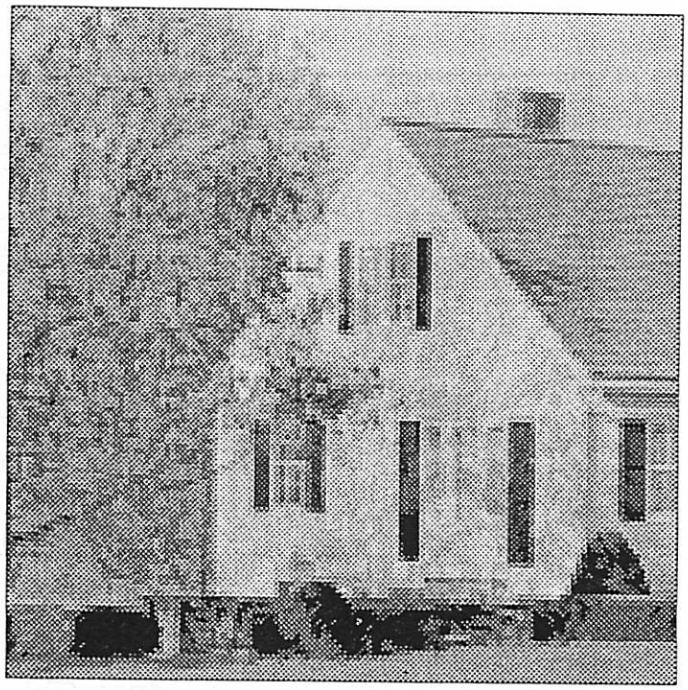


Figure 1.2: Intensity Image of Outdoor Scene

ing object boundaries in the segmentation produced by the clustering algorithm, resulting in the class of segmentation error known as overmerging. An example of this class of error is the lack of a boundary between the sky and house wall in Figure 1.4.

Existing image interpretation systems attempt to compensate for these two types of error within the high–level interpretation process. Undermerging errors are typically corrected by grouping processes which are used to merge adjacent regions with similar characteristics or labels [183,203] . Overmerging errors are typically bypassed implicitly through the use of a segmentation which has such a fine resolution (i.e. an overfragmented segmentation) that it can be assumed that any object boundary that does exist in the image has a corresponding region boundary in the image [74,122,148].

However, since both types of errors may occur simultaneously in different areas of a given segmentation, it is reasonable to assert that at least some forms

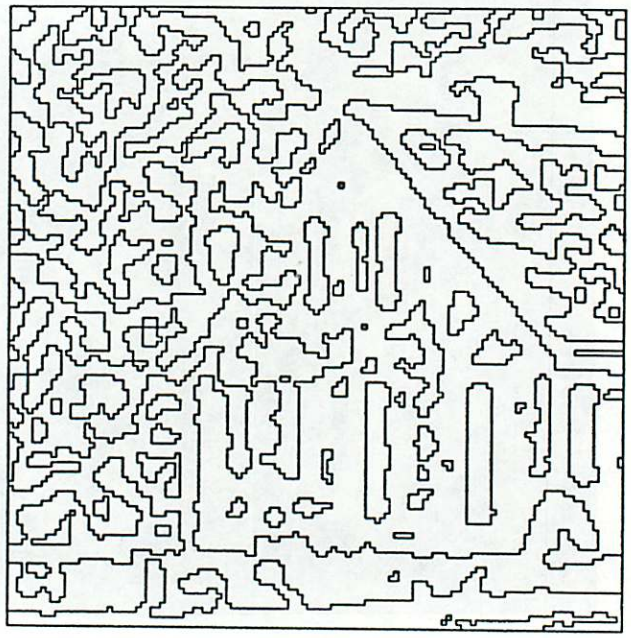


Figure 1.3: Zero–Crossing Segmentation

of error correction properly belong within the domain of the low–level segmentation processes themselves. As an interpretation process discovers problems with a segmentation, the segmentation itself should be redefined to resolve those problems. This redefinition of the segmentation may require the application of a different segmentation algorithm, or of the same algorithm with different sensitivity settings, over the portion of the image which presents the difficulties for the interpretation process.

Another factor leading to variability in low–level segmentation processing has to do with the *choice of image features* over which the segmentation processes will run. In typical multispectral (e.g. RGB) images, a large number of computed image features can be defined, some of which are known to be effective in segmentation processing [134,147,148,176]. However, in many segmentation algorithms, only intensity (the average of R, G, and B) is used. In other algorithms, the selection of features is typically made in an *ad hoc* fashion, where the general

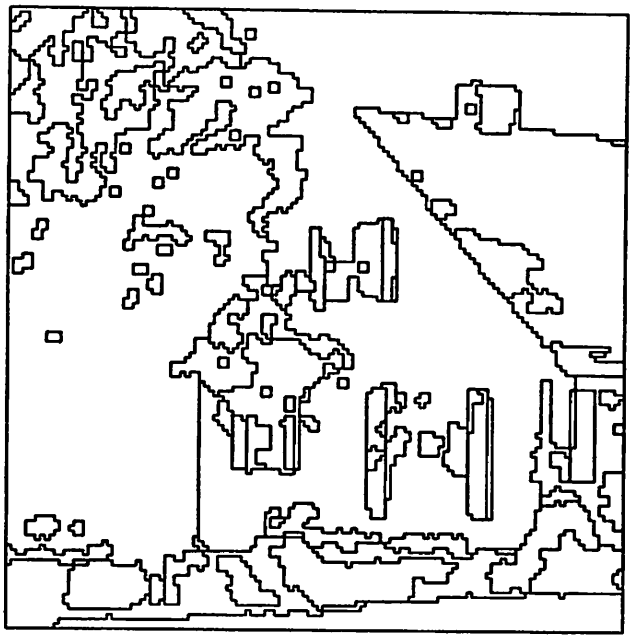


Figure 1.4: Histogram Clustering Segmentation

experience of the person designing the segmentation algorithm is brought into play. If such knowledge and experience were represented explicitly in the system, it would be possible to perform this selection automatically in response to the overall goals of the interpretation process.

Finally, there is the problem of *evaluating* the tokens of different region/line segmentations for their reliability or integrity with respect to the image data. Despite some efforts to quantify the evaluation of segmentation processing [77,140,149], it has been our experience that no low–level evaluation measure restricted to making measurements on the segmentation can provide a useful comparative metric. Rather, the quality of a segmentation can be measured only with respect to the goals of an interpretation process which uses the segmentation data.

In view of this variability in the selection of different segmentation methodologies and the difficulties of evaluating the results of various techniques, it becomes apparent that some form of intelligent intermediate–level control over the low–

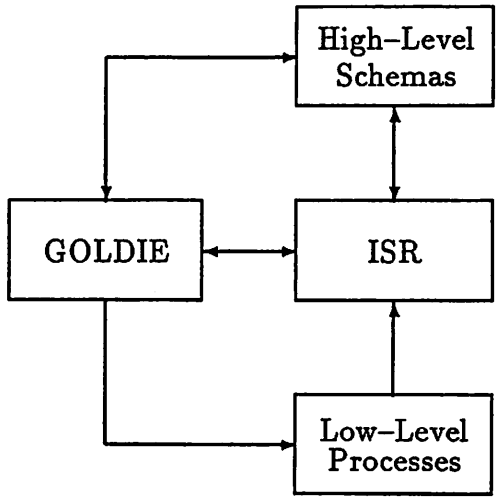


Figure 1.5: Position of GOLDIE in the VISIONS Hierarchy

level processing is necessary.

1.4 Design of GOLDIE

In order to construct a system that could provide intelligent intermediate–level control, a system called GOLDIE was designed (Figure 1.5) to function within the existing model of image interpretation realized in the VISIONS system (Figure 1.1). By addressing the issues of control for low–level segmentation processing, representation of intermediate–level data, and the control of intermediate–level processing, this module provides the framework through which this intelligent control can be exercised.

1.4.1 Segmentation Processing

As was mentioned previously, the current state of the art in segmentation processing encompasses a tremendous range of techniques [92,170], but little consensus over which are the most useful or when they should be applied. Methods such as region-growing, cluster labeling, thresholding, edge detection, line extrac-

tion, and surface reconstruction have all been found to be effective within limited domains, but no general methods have been found which perform adequately across a wide variety of images. For the most part, this lack of generality appears to be the result of the global application of local segmentation operators, rather than a failure of the specific techniques.

In view of this diverse nature of segmentation techniques, it becomes apparent that it is necessary to have some form of intelligent intermediate–level control. The view presented in this work is that the most effective form for the control of segmentation processing is the selective application of a set of segmentation algorithms to an individual image such that the most effective algorithm is chosen to operate over those portions of the image for which it is most appropriate. In other words, the overall process involves the formation of a hierarchical set of segmentation plans in which the results of early processing are combined with the intermediate results of the interpretation process and used to guide and constrain the later segmentation processing.

Such a mechanism permits the application of semantic or goal knowledge when it is available, but can also function in an environment where no such knowledge is available. We have therefore developed a hierarchical segmentation executive, using schemas as the basic control structure, to control the application of the various segmentation algorithms to an image in a consistent manner. The segmentation processes are applied over a selected portion of an image, and are capable of operating in parallel.

In most of the existing work on region–based segmentation, it is assumed that either the pixels represented within a region can be shown to exhibit a certain degree of homogeneity with respect to the image data or that they can be shown to be different from the pixels in adjacent regions. This distinction does not necessarily hold in the GOLDIE system. In this system, a region is defined to be a connected set of pixels which exhibit some desired characteristic with respect to the interpretation process. The system is not restricted to

predefined segmentation criteria, but is allowed to produce regions for which the definition of homogeneity is dynamically selected as a function of the state of the current interpretation process. This definition of what constitutes a region allows an interaction between the interpretation and segmentation processes that can be used to greatly increase the quality of resulting image abstraction.

The GOLDIE system implements this control of local segmentation processing through the process controller (described in Chapter 3).

1.4.2 Data Structures

Much of the flexibility of the GOLDIE system has to do with the intermediate–level data structures present in the system; these structures serve as the interface between the classically separate segmentation and semantic interpretation levels. The primary function of the data structures is to provide a representation for image tokens that is consistent at all levels of the abstraction hierarchy. However, the system must also provide a consistent representation for image data, hypotheses, control knowledge, hypotheses and state information so that a variety of low and intermediate–level processes can be allowed to share and operate on this data.

With respect to the token data structures, the two types of tokens which are most significant for the current implementation of GOLDIE are *lines* and *regions*. A line represents some form of discontinuity in the image, and is represented through a pair of image coordinates indicating the endpoints of the line. Typically, these line abstractions are not assigned semantic identities; instead they may be grouped with other lines and regions to make up an object or they may be used to guide other segmentation processes.

As was noted earlier, a region is defined to be a connected subset of the image pixels which exhibits some form of homogeneity with respect to the image interpretation process. The underlying justification for the region–based representation is that regions should correspond in some manner to the projections of

distinct objects in the image. The region segmentation process can be viewed as one which reduces the raw data to a set of entities which are believed to be descriptive of the data contained in the image. If the derived regions closely match the projections of objects in the image, the image interpretation process becomes more focused, and thus the processing becomes more efficient and reliable.

Because this is a system for research into image interpretation, we require a data representation that is flexible and extensible. We must have the capability to experiment with new image features, classes of hypotheses, or methods of data abstraction in an immediate and interactive basis. Chapter 4 provides a detailed description of the structures that have been implemented in GOLDIE to provide this data representation.

1.4.3 Knowledge Representation and Control

The complexity of the image interpretation task requires that we address a variety of control issues in the GOLDIE system. Foremost among these is the issue of the representation for knowledge and control. We require a representation for information about processing state that will allow consistent control throughout all of the levels of the system. As such, this representation must be highly modular and have the ability to represent complex dynamic interrelationships between structures, hypotheses and knowledge.

The control mechanisms must also support a goal–directed hypothesize-and-test protocol. The concept of a goal is central to an understanding of the interpretation and segmentation processes in GOLDIE: *interpretation has meaning only with respect to a set of goals.* A system which would spend a significant amount of time in the precise identification of tree texture might be poor design for a navigational system (where the goal is to find roads and obstacles), but would be reasonable if the goal was to identify foliage. Thus, it is clear that we require an explicit representation of interpretation goals. These goals may either be static, prespecified goals, as in the case of the foliage identification system, or they may

be created dynamically as subgoals to some other system goal. Goals of this form allow segmentation processing to be directed locally for maximum utility and also support focus–of–attention processing. By identifying critical subgoals of the interpretation process, it becomes possible to direct the system to address these specific goals.

The control processes must also support the parallel nature of image processing. Although processing is currently performed on a serial machine, we attempt to simulate parallel operation whenever possible under the assumption that we may, at some point in the future, be able to utilize true parallel processors. This parallelism takes two different forms in vision processing. At the sensory level of processing, where we have image processing and segmentation processing, there is no interaction between the processes occurring at different locations and the parallelism is primarily spatial. At the intermediate and high level of the interpretation process where we have a variety of cooperative processes acting on different areas of the image the parallelism is primarily temporal. The former is uniform across the image and synchronous, while the latter is non–uniform and asynchronous.

We have developed the schema control mechanism in response to this set of criteria. Schemas are modules which represent the set of strategies which can be expected to satisfy a particular goal. As goals are posted on the goal blackboard specific schema are instantiated to satisfy the goal. Chapter 5 provides a discussion of these issues and of the implementation of the schema mechanism.

1.5 The Structure of GOLDIE

An overview of the major components of GOLDIE that will be discussed in this dissertation is summarized in Figure 1.6. The inclusion of the interpretation schema instantiations in this diagram illustrates the interface between GOLDIE and the interpretation system; the remaining six structures represent the top–

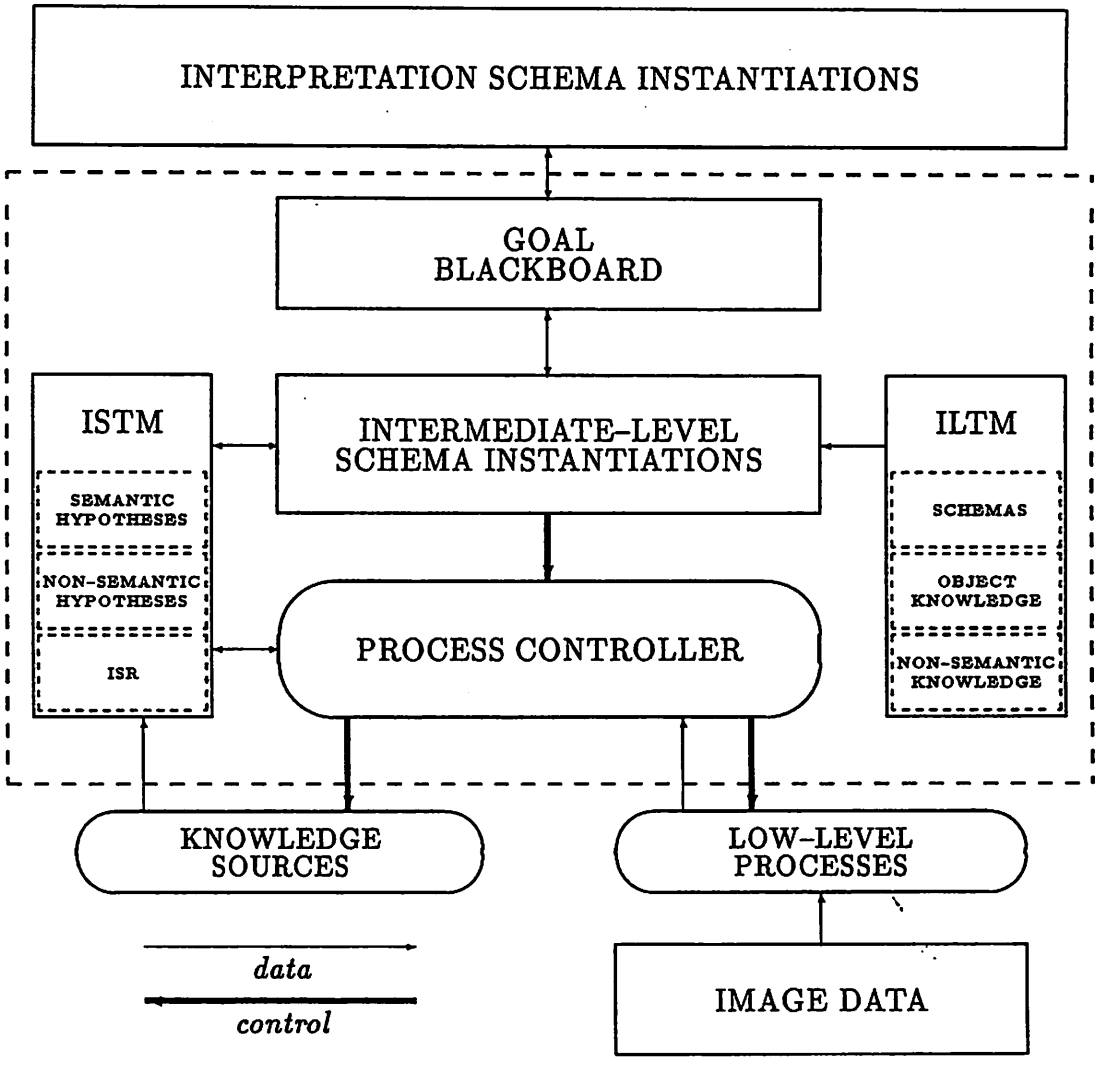


Figure 1.6: Overview of the GOLDIE System

level components of GOLDIE itself. The rectangles of this figure describe the components used for representation, while the ovals describe processes that operate on the data. Each of the modules is implemented as a LISP package, and all communication between modules is accomplished through a set of explicit communication protocols. It has been an underlying design philosophy that all such components of the GOLDIE system have an explicit representation and set of communication protocols so that all operations of the system are modular and reasonably free of side effects. Additionally, this explicit representation allows all interactions within the system to be traced and recorded, making it possible to develop an understanding of the complex operation of the system.

1.5.1 Image Data

The Image Database is the storage structure for the image data of the system. The underlying computational structure is that of a *Processing Cone* [81] which is a general, hierarchical, and parallel data structure for the representation and processing of image data. This structure is organized according to the concept of "levels", where the representation of an image at level $n-1$ has a resolution in both the *row* and *column* dimensions which is half the resolution of the image at level n (Figure 1.7).

As an image is brought into the processing cone, it is placed at a level corresponding to its spatial resolution. Other levels of the cone are then constructed through the application of reduction or projection operators to the original data. This data representation provides great flexibility in the way the image data can be viewed. Initial segmentation processing can occur over a reduced resolution version of the original image to provide an inexpensive plan for further processing, exploiting the redundancy of information in the image. This reduced resolution view of the image data also provides the capability to estimate both syntactic and semantic content of the image, providing information which can be used to guide the processing of the full resolution image. Additionally, this structure provides

1.5.5 Goal Blackboard

The goal blackboard encodes the control state of the interpretation system. The structure is somewhat similar to the blackboard structure of Hearsay II [56], and is a representation of specific goal that the schemas of GOLDIE are attempting to satisfy. This explicit representation allows a system to operate according to the goal-driven protocol, and serves as the communication pathway between the independent schemas of the system.

1.5.6 Intermediate–level Schema Instantiations

The Schema Instantiations represent the set of active schema–goal bindings in the system. As a schema is activated in response to a particular goal by creating an instantiation of the schema, it is bound to the goal via a set of parameters derived from the goal.

1.6 Environment

The GOLDIE system is implemented in LISP which runs as the top level of the **VISIONS** Image Operating System (IOS) [113]. The IOS is an integrated environment which provides a mechanism in which image operations may be performed on arbitrary image data under LISP level control. It allows the low–level processing to be performed in a single unified environment in which the symbolic levels of the interpretation process may interact in a straightforward manner with the low–level image processing. The implementation of the IOS is discussed briefly in Chapter 5; the reader is directed to [115] for a complete description of the system.

The image domain for this dissertation is primarily a set of complex images of natural scenes. These images contain a wide variety of natural terrain and man–made structures, where the modularity and goal–directed behavior of the system may be demonstrated with great effectiveness. The input data consists

solely of multispectral information (i.e. red, green, blue); range information from active sensors (e.g. laser range finders) or passive analysis (e.g. stereo or motion), are possibilities for future extensions.

1.7 Experimental Protocol

The GOLDIE system has the capability to operate in both data–directed and interpretation–driven modes. Data–directed processing is used to create a set of tokens that satisfy an intermediate–level view of the overall processing constraints (e.g. *"segment an area of the image to produce region tokens that are homogeneous in texture"*). Interpretation–directed processing, on the other hand, is used to redefine a set of tokens to better serve the needs of an interpretation process (e.g. *"resegment an area of the image to produce a region token(s) which may be interpreted as a bush"*).

Although a system does exist in the VISIONS research community for scene and object interpretation, it was difficult to directly integrate that system within the scope of this dissertation due to the changing and evolving nature of the system. Consequently, the set of goals for the experiments to be demonstrated in Chapter 6 have been specified through user interaction. The approach to this goal specification has been to identify those problem situations which have actually caused difficulty for the interpretation process, and to duplicate these problem states within the context of the GOLDIE system. Through the cooperation of many other researchers in the VISIONS research group, a more than sufficient number of these problem situations have been discovered.

1.8 Contributions

The primary contribution of the system presented in this dissertation is that it allows goal–driven control over the low and intermediate–level processes that produce or modify the tokens used in image interpretation. GOLDIE provides

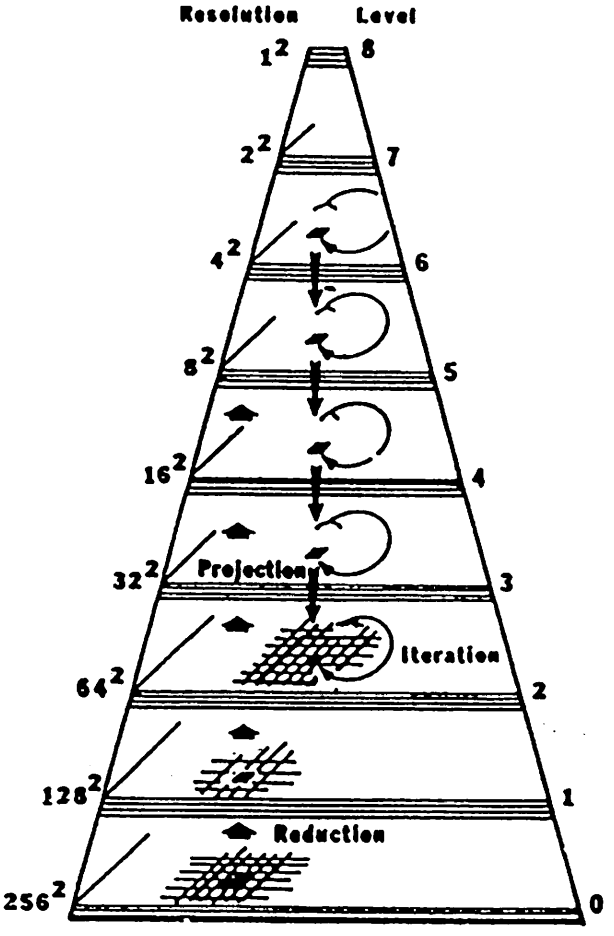


Figure 1.7: The Processing Cone

an efficient implementation of many low level processes which make use of the local context of a pixel [35,81,97,98,155,164,182].

1.5.2 Process Controller

The controller maintains a consistent protocol for the activation of both the low–level processing tasks that operate on pixels (e.g. threshold segmentation, one or two dimensional histogram–based segmentation, feature calculation, line extraction, etc.) and the intermediate–level knowledge sources that operate on tokens (e.g. collinear line grouping, token evaluation, or region merging). This interface permits the schema representation of these processes to be expressed in a manner independent of the process implementation. Note that the processes themselves (both low–level image–based procedures and intermediate–level knowledge sources) are not considered to be an integral part of GOLDIE; rather they represent a potentially extensible set of tools available to the system.

1.5.3 Intermediate Long Term Memory (ILTM)

This data structure encodes the image–independent intermediate–level knowledge of GOLDIE. This knowledge includes schemas, intermediate–level semantic object knowledge, and non–semantic knowledge regarding methods for the intermediate–level evaluation of image tokens.

1.5.4 Intermediate Short Term Memory (ISTM)

ISTM represents the image–dependent dynamic data structures of GOLDIE. Although ISTM overlaps with the STM of VISIONS for the storage of image tokens in the ISR, this module also maintains information about the relations between tokens, a history of the processes which produced tokens, image feature data, and hypotheses about tokens.

a mechanism by which the image interpretation process can "tune" the lower–level segmentation and grouping processes in order to produce tokens appropriate to a specific interpretation task. Through the interaction between processes at different abstraction levels, the quality of data produced by each process, and therefore the quality of the overall interpretation process, can be improved. This interaction is made possible through a representation for data and control that is consistent across the three levels of processing. The schema representation allows a coherent specification of the control strategies that are used at each of the processing levels, and the intermediate–level data structures allow the various processes to communicate with a consistent set of data primitives.

The explicit knowledge representation embodied in GOLDIE provides a framework for the declaration of information about tokens and also about the various processes that produce those tokens. This representation allows intermediate–level schemas to select specific processes that are appropriate both to the goal constraints and to the characteristics of the tokens being processed. Rather than using an approach in which fixed, global, data–directed segmentation processes are used to produce a static set of intermediate–level tokens [92,115,136,137,140,148], GOLDIE makes use of the goal–directed paradigm to execute a variety of dynamic and varying strategies to extract image tokens from the sensory data.

GOLDIE is not restricted to the use of a single segmentation algorithm. A set of segmentation algorithms (some of which had been developed previously and some of which are newly developed) have been combined into a single unified low–level control system. This control allows us a choice in the selection of the appropriate algorithm for a particular situation.

Chapter 2

Review of Relevant Literature

2.1 Introduction

The general topic of goal–directed control in image interpretation covers a wide range of issues and techniques. In order to provide a background for the topic, it is necessary to review a wide variety of work on segmentation, interpretation, artificial intelligence, knowledge representation, and control. The breadth of the topic also precludes an extensive review of all significant research in each of these fields; instead, selected examples will be presented which indicate the types of problems which can be faced in image interpretation.

This chapter will discuss the types of low–level (and sometimes intermediate–level) processing that can be used to create tokens for interpretation processes. The purpose here is not to provide details of implementation, but rather to provide a context for the research described in the following chapters. The specific segmentation algorithms utilized in the GOLDIE system will be discussed in greater detail in Chapter 3.

The organization of this chapter will approximately follow that of Section 1.4, which described some of the difficulties involved in segmentation processing. We will begin with a discussion of the characteristics of digitized images. This will be followed by a discussion of current techniques for image segmentation, both region–based and edge–based. Region–based techniques are those which use a grouping process over image pixels to produce regions: connected sets of pixels which share a common intermediate level label. Edge–based techniques, on the other hand, use a representation of discontinuities in the image as the significant primitive, and produce elements which are straight lines between endpoints.

The final section, which reviews knowledge–based segmentation systems, will address the issue of the evaluation of segmentation data and will discuss methods that have been proposed for the representation and use of both high and intermediate–level knowledge to influence the segmentation process. In order to provide consistency to this discussion, we will use the term *tokens* to represent the symbolic entities that could be constructed from segmentation data (e.g. region label planes or edge maps), regardless of whether or not the particular technique actually produced symbolic output.

Throughout this presentation, the focus is on the wide variety of algorithms that are potentially available to an image interpretation system. Most of these processes can be used to produce valuable data, but only if the goal of processing is matched with the strengths of the algorithm being used. By providing an overview of the strengths and weaknesses of each type of approach, the stage is set for the remainder of this dissertation in which we demonstrate how GOLDIE matches goals and algorithms in the attempt to produce the most effective data for any given goal context.

2.2 Image Representations

The digital representation for an image is a discrete array of values, or pixels, representing the strength of the input signal reaching the sensor from a set of selected (or sampled) points in the original scene. In many cases, this array is a digitized representation of a video signal. Ballard [10] presents a comprehensive study of the mechanisms by which the video signals are processed to form the image.

The characteristics of the digitization process introduce several difficulties with respect to the segmentation and interpretation processes. Since these images are discrete representations of a continuous phenomenon, significant errors can be introduced by the digitization process. Edges which exist in the scene

become blurred, or aliased, during digitization, and continuous ranges of intensity become quantized. These effects introduce a great deal of uncertainty into the segmentation process at the lowest levels. Without a global perception of context, the effects of aliasing and quantization often make it impossible to form a valid interpretation of the significance of the value at an individual pixel (*cf.* Section 1.4).

2.2.1 Color Space

The preceding definition of an image does not restrict the representation of an individual pixel to a single scalar value. Often, a pixel may be represented by a vector of scalar values representing the intensity of various types or bands of radiation.

Color space is a method of organizing the representation of spectral information in the image. Since most visible colors can be represented and reproduced by a three point frequency sampling, this has become a widely used representation for color space. The most commonly used sampling points correspond to the red, green, and blue (RGB) frequency components of the composite color. The RGB representation assumes that the intensity values in each of these three colors lie along the three mutually orthogonal axes of a three dimensional space. A great deal of analytic study has been made of this color space representation [189], and the information provided through this representation can be quite useful for a variety of interpretation tasks. For example, classification systems have been developed which utilize statistical pattern classification techniques over this space to measure land use and agricultural data in aerial imagery [66,132].

A major difficulty with the RGB representation is that the red, green, and blue axes of this space are not actually mutually orthogonal. In natural scenes, where color saturation is low, these features are typically highly correlated. Therefore a number of techniques for image segmentation systems make use of other color basis points to define the color space [158]. Intensity, Hue, Saturation (IHS)

[110,176], or YIQ [147,149] may at times provide image representations that are more useful to segmentation and interpretation processes (*cf.* Section 2.3.1.3). Other systems use a dynamically computed set of image features that are guaranteed to be statistically independent. For example, Coleman [43] uses a Karhunen-Loeve transform to compute a linear transformation of the data into a basis vector of independent features. By computing the transform over a multidimensional feature set that includes: red, green, and blue, (as well as the non–color feature representing the log Sobel edge magnitude), this system dynamically computes a new basis for color space. Measurements of the frequency distributions of these computed features are then used to select effective features for image segmentation. The segmentations produced according to these various features varied dramatically according to the specific image feature chosen.

Given the evidence provided by these and other bodies of research, it is clear that the problem of choosing a "good" set of features to represent an image is far from solved. It may be that there is no "good" set in a global sense, and that the choice of features is dependent upon the goal of the system. In this dissertation, we adhere to this latter view, and consequently the choice of appropriate image features is one of the parameters manipulated by GOLDIE (*cf.* Chapter 3).

2.2.2 Texture

One of the most troublesome characteristics of natural images is texture. Rather than being a property of an image which may be directly measured at each pixel, texture is defined by the relationships of pixel values over a local neighborhood. We may think of image texture as the close juxtaposition of relatively small events in the image. Texture may be regular, as in the case of a brick wall, or may be highly irregular, as in the case of tree foliage. The problem with texture is increased when the individual texture elements, or *texels* [108], become small enough that it is difficult to recognize them individually. A significant body of work demonstrates the difficulty in the identification and classification of texture

in images [40,73,85,118,153,207].

Haralick et. al. [89] provide a survey of global texture measures (i.e. measures for which the neighborhood is defined as the entire image) that can be used to classify areas of aerial images according to textural measures. These measures represent statistical abstractions of the directional cooccurrence matrix for intensity values over the image. Measures such as Angular Second Moment, Contrast, Correlation, and Variance are used to demonstrate an 80% correct classification rate over their set of 314 test samples. Categories such as coastal forest, woodlands, and urban areas are used as labels for the classification process.

The segmentation of images based on textural information is more difficult than simple classification, however. In order to measure texture, statistics must be measured over an area, but the nature of this process will tend to spatially blur the image feature data. As the local neighborhood becomes larger, the measurement of textural characteristics becomes more accurate at the interior of a textured object, but less accurate near the periphery. Van Gool et. al. [73] call this the *window problem*:

> The fundamental problem is, again, the choice of appropriate window sizes. When texture features are measured on small subimages, they are unreliable, but if we use large subimages, it is hard to find regions that are uniformly textured. On the other hand, if we want to segment an image into differently textured regions, the blocks will have to be as small as possible to enable accurate location of the borders between the regions without a blocky outlook.

The approach that has been used in GOLDIE is to use relatively small window sizes in the calculation of textural image features so as to be able to locate the boundaries of textured areas with a reasonable degree of precision. As will be demonstrated in Chapters 3 and 6, the use of the image features for segmentation purposes may be constrained in such a manner that the statistical unreliability of the small windows does not present too great a difficulty.

Although not directly related to the research described in this dissertation, it is worth noting that textural information can be used at a more semantic level to directly determine characteristics of objects in the image. Kender [108] demonstrates the use of texture information in the determination of shape. A significant body of work [46,110,201] extends this original thesis, and demonstrates how the straightforward measurement of textural characteristics can, under some conditions, extract tree–dimensional surface properties from images.

2.3 An Overview of Segmentation Techniques

There are almost as many segmentation techniques in the literature as there are researchers in low–level vision. Broadly, however, theses techniques fall into three general classes: region-based techniques, edge-based techniques, and techniques which make use of a combined region/edge representation. In this section, we will provide a brief outline of a variety of techniques within each of these classes, and at the same time explain the types of situation in which they can potentially produce useful segmentation data, as well as situations in which they will often fail to produce useful data. The point here is to demonstrate the variety of techniques that are available for image segmentation and to show the way in which explicit or implicit intermediate level knowledge is utilized in the execution of these segmentation tasks.

2.3.1 Region–Based Segmentation Techniques

As the name implies, region-based techniques utilize regions, or sets of pixels, as the basic primitive. As originally specified by Brice and Fennema [26] the objective of a region segmentation process is to *group* sets of similar and adjacent pixels into region tokens. The factor that differentiates the various region segmentation techniques is the method by which this similarity is established.

2.3.1.1 Region Growing Techniques

In region growing techniques the pixel grouping criterion is highly local; similarity between pixels is established through pairwise relations on all sets of spatially adjacent pixels in the image. In the simplest case of region-growing, a seed point is first established at an arbitrary pixel which is then given a unique label. Each spatially adjacent pixel whose image feature value lies within some threshold of that of the seed point is given the same label and then becomes a seed point itself. When no more pixels can be added to this region, an arbitrary unlabeled pixel is chosen as a new seed point and given its own unique label. The process continues until all pixels have been labeled.

On a more conceptual level, each pixel of the image may be represented by a node in a graph and neighboring pixels whose properties are "similar" enough are joined by an arc. Regions are maximal sets of pixels all belonging to the same connected component [68,85]. In one demonstration of this approach, Gambotto and Mongo [68] describe a system in which the set of adjacent region pairs with the lowest absolute difference in gray levels are merged at each iteration. Processing stops when the the gray-level difference between all adjacent regions exceeds some maximum threshold.

However, similarity need not be defined strictly by the difference in image feature values. Bracho and Sanderson [24] use a region growing algorithm based on gradients to segment curved shaded surfaces. In this case, the criterion for region growing is not the difference of intensity values between pixels, but rather the difference between the local intensity gradients which have been measured at each of the pixels. Pong et. al. [157] segment aerial images using a similarity criterion that the two adjacent pixels must appear to belong to the same polynomial intensity surface.

What these region growing algorithms all share is the problem of *leakage*. Leakage describes the situation in which highly dissimilar areas become connected

through a small group of pixels which form a low-gradient *bridge* between the two areas. Since the decision to grow is always based on a *local* pairwise relation, no state information is available about the characteristics of the other pixels which are already contained in the region.

In order to combat this problem, another category of segmentation algorithms has been developed, which Haralick [92] calls hybrid linkage combination techniques. In this category, the pixel nodes possess a variety of attributes based both on pixel intensity values and characteristics of the local neighborhood of the pixel. By using edge information [87], or local surface fits [86], the region growing process may utilize local context, and demonstrate more reasonable behavior.

Levine and Shaheen [122] utilize a process in which the image is scanned in raster order, with pixels being grouped if their differences in each of red, green, and blue are less than a threshold. This threshold is determined dynamically as a function of the mean and standard deviation of the current region. Through this use of multiple features and adaptive thresholds, the technique inhibits, to some degree, the problem with leakage. By using a global view of the data across the existing region, the local decision about whether to merge an individual pixel can be made more reliable.

A more sophisticated approach to region growing is proposed by Griffith et. al. [18]. In this two stage process, region growing first occurs at the pixel level, using similarity in image feature values, and then occurs on the level of regions themselves, growing new regions over the set of original regions. The initial set of regions is created by a very conservative region growing algorithm which (in the natural scene domain) leads to a segmentation with a very large number of regions. At this point a rule–based region merge algorithm is used to iteratively merge adjacent regions based on the similarity of *region* characteristics. The rules themselves incorporate a form of intermediate–level knowledge about image characteristics. Some rules are quite simple, representing facts such as "adjacent regions with similar feature means (e.g. intensity) should be merged", while

others are more complex, representing information such as "the resulting region should have a feature variance below some threshold". Thus this algorithm actually operates on both the low and intermediate levels. The initial set of regions is formed through a low–level region growing process, while the final segmentation is defined through a region–merging process over the set of intermediate–level region tokens.

The most important characteristic common to all of these techniques for region growing is that they make decisions based solely on local (pixel or region) characteristics; there is no use of more global information about image context. This is in strong contrast to the split–and–merge techniques which initiate processing at a global level and iteratively become more localized.

2.3.1.2 Split–and–Merge

Split–and–merge techniques [38,67,98] work under a hierarchical model in which the initial image to be segmented is a reduced version of the original data. Often this hierarchy is organized spatially, and is arranged as a quadtree [172], or as a pyramid similar to the processing cone [81].

Given a region labeling of the image at any level of this cone, all pixels at the next finer level of resolution are given (inherit) the label of their ancestor in the hierarchy. Each region is then examined for homogeneity, which generally means that the variance of feature values for the descendants does not exceed a threshold. If the pixels which comprise the region are not homogeneous, the region is split into several new (smaller) regions. Once all splits have been made, a merging pass is used to merge adjacent regions which are similar in their characteristics.

By starting with a one–pixel region at the top of the processing cone and iterating this process down to the level of resolution of the original image, a complete labeling of the image is produced. The measure of homogeneity that is used to make a decision whether or not a region should be split need not be based on simple variance. Horowitz and Pavlidis [98] describe a split-and-merge

procedure based loosely on the assumption that the distribution of intensity values across a region was Gaussian. Given an image, a pyramid is constructed in which the pixel values represent the variance of the values of the descendants of the pixel in the pyramid. As the split–and–merge process iterates down the hierarchical structure, these variance measures are used to determine homogeneity.

Cohen et. al. [40] present a parallel, hierarchical split–and–merge segmentation algorithm that is based on a Markov random field model. The basic approach is similar to that of the general split-and-merge, but in this implementation each region is evaluated under the assumption that it represents a collection of two Markov fields. If any element of the pyramid shows evidence of two classes of texture, it is partitioned until only one class appears.

Variations to the basic control model can also relax the rigid conditions of homogeneity in the determination of whether to force a split. The technique of Hong and Rosenfeld [97] makes use of an iterative relaxation scheme based on the spatial hierarchy of the pyramid. The pixels of a given level of the pyramid are linked to their ancestors through a set of weighted links describing their similarity to their siblings. Image values are recomputed as a combination of weighted ancestor values and original values. This process iterates until a *stable* solution is found, at which time a connected components process is used to produce the final segmentation.

Aside from the obvious efficiency which can be derived from the parallel nature of these techniques, the top–down approach that they exemplify can also be extremely valuable in the abstraction of the spatial distribution of image content. In essence, the first few iterations of region splitting form a gross plan for segmentation, in which significantly different areas of the image are identified. However, the valuable information that could be gained from this segmentation plan is never used. The same form of processing occurs at each level of resolution: only the spatial resolution and feature values vary. Thus, in contrast to the region–growing techniques that did not make use of global information, these

techniques fail to make use of local information to constrain the segmentation process. These techniques do not make use of the characteristics of the regions from the initial plan to dynamically adjust the split–and–merge thresholds to match the characteristics of the original data. A choice of thresholds or image feature that works well in one image may fail entirely in another. If a texture feature is chosen, smooth image areas will be overmerged; if a color feature is chosen, textured areas will likely be overfragmented. These techniques may also fail to identify regions that are obvious to a segmentation operator that has a more local view of the data. Thin regions that are obvious in the original data, but which happen to span several of the reduced resolution regions, can easily become fragmented or missed entirely.

2.3.1.3 Measurement Space Clustering

Measurement space [92] guided spatial clustering involves an analysis of some measure of the image to find a basis for segmentation. The basic methodology is to identify spatial clusters within some measurement space[1] under the assumption that these clusters represent one or more sets of similar (relative to the chosen measurement space) pixels which are spatially adjacent. The measurement space itself may be a simple one–dimensional histogram representing the frequency distribution of image feature values, or it may be a more complex multidimensional space that represents the spatial relationships between a set of image features. Given an appropriate selection of a measurement space and an appropriate selection of clusters in that space, it is possible to produce region segmentations that provide an appropriate level of abstraction for an interpretation process.

[1]In order to clarify the distinction between measurement and feature spaces, it should be noted that a an *image feature* is a specific array of data which has been computed from the input image data, and is in spatial registration with that data. A *measurement space*, on the other hand, is the (multidimensional) space representing the global distribution of (a set of) image feature values across the area of the image that is to be segmented.

The difficulty with this class of algorithm lies first in the ability to identify the appropriate measurement space for a particular segmentation, and then in the ability to correctly identify the clusters in that space. A further difficulty lies in the fact that the classification is based on a global abstraction of the image data expressed in the measurement space; no local information is used in the classification of an individual pixel. For this reason, many systems that use measurement space clustering techniques actually utilize a two–stage process to accomplish the segmentation. The first phase is intended to map cluster labels into the image data to provide an estimate of the segmentation. This estimate may be a form of probabilistic estimates of pixel labelings, or it may be an actual labeling. The second stage is a refinement step, in which the original estimates of region membership are updated using local context. This updating can be as simple as the removal of all small regions, or it may be a complex relaxation process [40,47,71,115,137,139,160,169,206] in which vectors of class membership probabilities at each pixel are iteratively updated in the context of the probability vectors of all neighboring pixels.

Thresholding is the most straightforward example of measurement space guided spatial clustering. In the most trivial case, thresholding is a technique for foreground/background classification in which a value is chosen as the threshold and all pixels with a feature value less than the threshold value receive one label while all with feature values equal to or greater than the threshold receive a different label. Typically the choice of this threshold is made by an analysis of peaks and valleys in measurement space [75,112], but the thresholds can also be chosen through an analysis of edge information in the image [114,130].

Kohler [114] suggests a different measurement space technique for segmentation that utilizes multiple thresholds selected according to global gradient information. Thresholds are selected so as to maximize the global average contrast of the region boundaries created by thresholding. The algorithm essentially computes the global average contrast for every possible threshold, and then selects

the threshold which produces the maximum boundary contrast across the image. The algorithm may be applied recursively by ignoring all edges which have contributed to previously computed thresholds.

A second class of measurement space segmentation techniques involves the simultaneous selection of multiple clusters in a one–dimensional space. This technique may either provide a set of multiple simultaneous thresholds [147,149], or may be used to estimate probabilities of membership between individual pixels and the set of clusters [54,137,139,169,170].

A third class of techniques makes use of clustering in a multidimensional measurement space. In this class of techniques, the feature space contains several dimensions corresponding to different image features [5,85,145,150,165,173]. Such an approach increases the potential for a precise classification of the individual pixels since more information is available, but the necessarily sparse nature of the feature space matrix (particularly when the technique is localized to subimages) makes the identification of clusters increasingly more difficult with the increasing dimensionality of the space. This type of technique is often used in the case of LANDSAT satellite images, where the measurement space is based on the frequency distributions across a set of image features representing six spectral bands extending from visible into the infrared that have been chosen for their utility in land use studies [31,198].

Measurement space guided spatial clustering can be used either for direct classification as above, or it can be extended to utilize the descriptions of measurement space clusters to form probabilistic estimates of pixel labels. An example of this latter approach was developed by Nagin [77,137], in the form of a one-dimensional clustering algorithm that determines peaks and valleys in the frequency distribution histogram of an image feature for the entire image. The peaks are used as the basis for a Bayesian estimator that gives a pixel a probability of cluster membership that is proportional to the distance in feature space between the pixel value and the peak that identifies the cluster. These probabili-

ties are updated through a relaxation process [169] that refines the labeling based on local context, and then a connected components process is used to produce the final segmentation.

The most unreliable aspect of Nagin's algorithm lies in the peak selection process. As was stated earlier, one of the most difficult problem in measurement space clustering is the precise identification of the clusters themselves. Because of uneven lighting, sensor distortion, and perspective effects, even ideal areas of the scene that have have constant color and reflectance produce a range of feature values in the digitized image. Typically the clusters will appear to be Gaussian distributed. Thus, if the cluster centers are close in measurement space, it becomes difficult to identify the valley between them. In many cases, a large cluster can completely obscure a smaller cluster.

For this reason, Ohlander [146,147,160] developed a hierarchical segmentation system that could spatially decompose the image and segment each portion independently, reducing the number of clusters in any given measurement space in the hope of lowering this interaction between clusters. In addition, the system is able to utilize a variety of measurement spaces, choosing the image feature with the best feature space characteristic in order to segment each of these local areas. This system was one of the first dynamic segmentation schemes, and although the system requires a great deal of interactive control, it represented a significant conceptual advance in segmentation research.

The overall segmentation process consists of three phases: preprocessing of sensory data, identification of texture regions, and recursive thresholding of the non-texture regions. Preprocessing involves of the calculation of a set of color features, which Ohlander calls parameters, that collectively represent a wide variety of views of the input data. Nine color features are used: red, green, blue, hue, saturation, intensity, Y, I, and Q (Table 3.2 in Chapter 3 provides the definition for these features).

The second phase of the process serves to isolate areas of texture that are

generally unsuitable areas for segmentation by thresholding. Texture areas are identified through the use of a *busyness feature* based on local gradients. A (3x3) Sobel operator (which produces a measure of local edge direction and strength) is applied to each of the nine color feature images. The resulting set of edges is processed to remove "true" (i.e. non–noise) edges; the local density of the remaining edges is calculated and thresholded to produce a *busyness matrix* that is used to construct a texture mask. This mask represents the regions of the image that will not be recursively thresholded in the third phase of the process.

The remainder of the image (i.e. the set of pixels not included in the texture mask) forms the current *template* for thresholding, and histograms for each of the nine color features are computed over all pixels in this template. Each of these histograms is then analyzed in order to determine the image feature that can be used to produce the best threshold. The selected threshold is applied, and each resulting region then became a new template for the recursive thresholding process. Processing continues until each of the templates (regions) is unimodal in all features, and therefore assumed to represent a primitive region[2].

This concept of spatial decomposition of the image is similar to the concept of *planning* proposed by Nagin et. al. [138]. According to this approach the first stage of the segmentation process is *plan generation.* The goal of the planning stage is to coarsely partition the image in order to reduce the complexity of the scene. It is assumed that regions will be overmerged in the plan segmentation, and that boundary positioning will not be exact. The regions of the plan segmentation are then resegmented in the *refinement* stage of processing. Since the information is more local at this stage, fewer clusters should be present in the

[2]Price and Reddy [161] and Shafer and Kanade [175] extended this work of Ohlander, and they suggested a variety of techniques that could potentially improve the process. Extensions which were proposed included: greater use of region characteristics for the determination of regions to be segmented, evaluation of region shape to measure threshold acceptability, and multiple thresholds across a single feature. All of these suggestions have been implemented in GOLDIE

feature space, and segmentation decisions may be made with greater accuracy. Thus the *segmentation plan* becomes similar to an intermediate level in the split–and–merge strategy, but in this case the boundaries of the segmentation plan are not low resolution approximations to the full resolution data.

This concept of hierarchical plan generation has been crucial to the development of region–based segmentation schemas in GOLDIE. Chapter 5 will present a detailed discussion of the way in which the concept of hierarchical plans for segmentation have been integrated with intermediate–level and high–level knowledge to produce a highly effective form of control for region segmentation.

In later work, Nagin [136,137] developed a segmentation system that utilizes the concept of local context for segmentation, but discarded the concept of hierarchical planning in favor of a more efficient parallel design. According to this technique, the image is partitioned into rectangular regions that are then segmented with a one–dimensional clustering technique in a (simulated) parallel fashion. The result of this process is a segmentation of the entire image that contains valid region boundaries as well as boundaries which are artifacts of the partitions. Methods developed by Nagin and Kohler [115] are used to communicate information between processes in adjacent sectors to permit the recognition of small peaks corresponding to clusters in adjacent sectors, and also to merge regions that had been arbitrarily split by the sector boundaries. This method makes use of an arbitrary partitioning of the image for localization, as opposed to the planning mechanisms of Nagin [138] or the recursive method of Ohlander [147]. Thus, although the localization provided by this approach is not as responsive to image context as is the planning model, the approach is well suited to parallel hardware [78].

2.3.1.4 Characteristics of Region–Based Techniques

The various region–based techniques that we have reviewed each have their own strengths and weaknesses. Region growing techniques are most appropri-

ate when there is no texture, and when all scene boundaries are represented by **continuous** and **sharp** intensity discontinuities in the data. Given appropriate measures of homogeneity, the split–and–merge techniques are able to deal with certain types of textured areas, but the hierarchical methodology for boundary formation can often lead to inexact localization of region boundaries. Measurement space techniques are highly dependent upon the nature of the image features chosen; an image feature that is useful in the discrimination of one particular class of object/region may behave poorly with respect to another class of object/region in the same image. The use of a hierarchy or a fixed spatial partitioning to localize the measurement space clustering may be quite useful in a complex image, but may introduce excessive computational overhead in a simple figure/ground discrimination.

Because of the modes in which each of the techniques can fail, none of these techniques are robust enough to serve as the sole means for the production of intermediate–level region tokens in a generalized interpretation system.

2.3.2 Edge–Based Segmentation Techniques

A conceptually different approach to image segmentation makes use of the differences between pixels, rather than similarity, to create image tokens. From this perspective, the objective of the segmentation process is to locate *edges* that represent significant discontinuities in the image data. Under the assumption that these discontinuities represent true discontinuities in the underlying scene, such techniques can be used to partition the data into identifiable elements. The value of this type of processing is strongly supported by a wide variety of psychophysiological and neurophysiological studies [3,6,11,100,101,116,119,125,126,127,128].

In the realm of computer vision, edge–based techniques may be used either to produce a set of regions that are bounded by continuous edges, or to produce a disconnected set of edges that provide information that is complementary to a region–based segmentation.

2.3.2.1 Edge–Based Region Segmentation

A number of techniques for the creation of regions via image discontinuities are based on the work of Marr [125,126,127,128]. According to this methodology, the initial determination of intensity discontinuities in the images uses a method modeled on the receptive fields of the human retina. Using the on-center/off-surround operator, Marr and Hildreth proposed an edge detector based on the zero-crossings of the second spatial derivative of the image intensity function [125]. Using of the digital Laplacian operator, edges are placed at locations where the intensity changes most rapidly (zero–crossing of the Laplacian). By varying the radius of the Laplacian operator, edges may be determined over a range of spatial frequencies.

Even though this algorithm is based on discontinuities in the image data, this algorithm is considered to be a region segmentation process in the discussion of segmentation algorithms that will be presented in Chapter 3. In the current implementation of GOLDIE, all edges are expressed as straight line segments, and thus the zero–crossing contours can not be placed in the line representation.

2.3.2.2 Edge Detection

If the objective of the edge–based segmentation is to produce a set of discontinuous lines rather than a set of closed contours, the edge segmentation process becomes less constrained. As is the case with region segmentation, research in the edge domain has produced a large variety of local edge operators that can be used to determine edges in images [17,48,55,84,86,90,93,95,96,103,130,179,185]. The most straightforward of these is a simple ($2x2$) difference operator that measures the difference between the intensity of pixels in a horizontal and vertical direction (or across the diagonals [168]). In this case, edge strength is some function of the two edge measurements, and edge direction is the arctangent of the ratio of the two edge measurements. Obviously, given the characteristics of digi-

tized images that were discussed earlier, a great number of the edges produced by this template are the result of digitization noise. One technique to remove these noise edges is simply to threshold the magnitude of edge data [65,130,156], but the problem of selecting the appropriate edge threshold is subject to the same difficulties as the selection of a region segmentation threshold.

All local edge detection algorithms assume that an edge may be found through the detection of a specific local discontinuity in the digitized data. This is not always the case, however. Often a sharp object boundary discontinuity in the scene will produce a slow gradient in the image, in which the effects of mixed pixels, texture, shadow, or boundary curvature cause the sharp scene discontinuity to be spread over two or more adjacent pixels. The difference in intensity values between the individual pixels thus represents only a portion of the total intensity discontinuity of the actual edge. More complex edge templates such as the Sobel [159], or Kirsch operators [111] use larger windows that makes the detection process much less sensitive to this effect at the expense of losing precise locality.

Several other researchers have developed edge detection algorithms that are based not on the local difference between pixel values, but rather on a continuous intensity surface that is constructed from those values. Grimson and Pavlidis [74] propose an edge detection method that uses the distribution of the error between a planar fit model of the data and the data itself. Using a statistical model of the Gaussian distribution of this error value, they demonstrate an algorithm that places edges at positions where the error distribution deviates significantly from the expected values.

Haralick [86,90] proposes a similar model for an edge measure, the *digital step edge*. By fitting a continuous surface (*facet*) over the fixed local neighborhood of a pixel, Haralick creates a mechanism through which it becomes possible to compute directional derivatives. Edges are created at gradient maxima that are determined through the selection of the zero-crossings of the second directional

derivative taken in the direction of the gradient.

A more generalized methodology for the design of edge detectors for arbitrary one–dimensional edge profiles is presented by Canny [36,37]. Working from a rigorous mathematical formulation of detection, localization and multiple response criteria for edges, numerical optimization techniques are used to provide the specification of the optimal operator. Other techniques such as adaptive local thresholding and nonlinear differential zero–crossing operators are then used to extend the methodology to actual two–dimensional image data.

A quite different model for the analysis of the intensity surface is proposed by Burns [33,34]. In this method, edges are first obtained by the application of a simple gradient operator to the image. The gradient directions are then quantized and a connected components process is used to form regions corresponding to groups of adjacent edge elements that have the same quantized **orientation.** A planar fit is made to the intensity surface over each of these regions. The line of intersection between this planar surface and the surface defined by the mean intensity over the region is then considered to represent a line in the image. The regions of this model are somewhat analogous to the facets of the Haralick algorithm, but since the size of each region is a function of local context, the Burns algorithm provides edges whose length is representative of the actual length of the edge in the image.

Although each of these "intensity surface" algorithms produces edge data that is usually quite accurate with respect to large structures in the image, texture or irregular gradients can still produce significant difficulty. The underlying assumption for this class of algorithm is that the intensity surface is typically smooth; edges are (relatively) rare events where the smoothness constraint fails to hold. If, however, the image data is highly textured, this assumption does not hold. In this type of data, it is the smooth areas of the intensity surface that becomes rare. This observation led to a different class of edge detectors that are sensitive, not to variation in the intensity surface, but rather to variation in textural

characteristics.

The difficulty with this model is in the representation of textural characteristics. In one attempt to define some of these characteristics, Haralick [86] reviews the statistical and structural properties of image texture. He reviews autocorrelation, textural edgeness, texture morphology and spatial gray–tone (co–occurrence) methods, and demonstrates the strengths of each. From the segmentation point of view, however, these methodologies all become ineffective, since they are all measures over a (relatively) large area around the pixel of interest. Haralick does demonstrate very good results for several of the methods in the classification of texture images, which would extrapolate to region–based interpretation classifiers.

Other texture classification schemes include the use of oriented line filters [193], Walsh transforms [75], combinations of different size spot detectors [185,207], autocorrelation [167], or the characteristic angle of the Fourier transform over local windows [32]. In each case, the goal is to identify some image feature which has a smooth intensity profile over image areas of constant texture. Edges are placed where the textural characteristics differ.

2.3.2.3 Segmentation from Edges

Once a set of edges has been detected in the image data, additional processing is typically required to produce an adequate segmentation. A single missing edge (e.g. resulting from noise or aliasing) in an otherwise connected set of edges can cause the region leakage that was discussed earlier. Since most edge detection processes produce only *local* edge estimates, these must be linked in some manner to produce longer lines. This implies the need for an intermediate–level process to *group* edges.

The grouping process can be complicated by a variety of factors. The set of local edges making up a line are disconnected, often with significant gaps resulting from the thresholding of weak edges, noise, lighting conditions, shadows, etc.

Even when the edge elements are linked into lines, these lines typically do not meet nicely at corners, T–junctions and the like. For these reasons, it becomes very difficult to produce a region segmentation from a set of edge elements or lines.

One of the earliest approaches to the grouping process was the Hough transform [99]. By parameterizing edges according to the slope and intercept of the equation of the edge, a two–dimensional measurement space representation of a set of lines is produced. Clusters in this measurement space indicate a set of edges that are collinear and may potentially be grouped. This classic measurement space approach to grouping has since been extended to perform grouping for a variety of edge structures other than straight lines [8,9,117,144].

Another early approach to grouping was that of Dudani and Luk [53], who use strictly local information in the grouping process. In this system, edges created with the Hueckel operator [102] are filtered to find high contrast edges, and then grouped on the basis of collinearity with other edge elements. An algorithm recently developed by Boldt that uses this type of local collinearity grouping [21,192] has been incorporated into GOLDIE, and will be discussed in detail in Chapter 4.

Hanson, Riseman, and Glazer [84] make use of edge continuity constraints in their approach to local edge-based segmentation. Initial edges for this system are created according to the horizontal and vertical differences in pixel values, and the gradient strength of the edge is converted into an edge probability. A relaxation process then updates every edge in the image based on the data contained in a small neighborhood around the edge and a local continuity model. Conditional probabilities for edge element pairs are determined according to theoretical analysis of edge continuity and gradient behavior. These conditionals are used in the Bayesian updating of the relaxation process to group edges into continuous boundaries.

Edge relations other than collinearity or continuity may also be used as the

criterion for grouping. Pietikäinen and Rosenfeld [154,155] use edge-based texture measures for image classification and segmentation. Initial edges are calculated with a ($3x3$) Kirsch operator. Two different edge grouping algorithms are then used, one that finds pairs of antiparallel edges, another that uses curvature characteristics of the determined edges. Global parameters are measured over these primitives, (e.g. mean distance between edge pairs or contrast difference between edge pairs) and used for classification. A segmentation is obtained through a pyramid–based nearest neighbor algorithm that operates over these edge characteristic measures.

2.3.2.4 Characteristics of Edge–Based Techniques

These examples of edge–based techniques once again point out the diversity of approaches that can be taken in the creation of intermediate–level tokens. The choice of which approach to use is a function of the goal of processing. If closed contours that approximate the spatial location of strong image discontinuities are desired, a zero–crossing algorithm may produce the most useful results. However, as will be demonstrated in Chapter 3, the requirement that this algorithm produce closed contours often leads to poor segmentation results in areas of slowly changing gradient. If, on the other hand, the set of long straight lines are desired, local edge detection and grouping would be more useful.

The choice of which image features to use is also critical. An edge detector that used an autocorrelation mechanism to locate texture boundaries would be quite ineffective at locating the edges of a picket fence. Even the grouping criteria are dependent on the goals of the system. Antiparallel line grouping would not perform adequately for locating road boundaries, while collinear grouping would be ineffective in the determination of foliage contours.

Although it is sometimes not expressed clearly in the presentation of these edge algorithms, each of these edge algorithms is designed with an implicit intent or set of assumptions. The utility of the algorithm in a vision system depends

upon how well these assumptions are satisfied with respect to the processing goals of the system.

2.3.3 Techniques Using a Combined Region/Edge Representation

In view of the inherent difficulties with both edge–based and region–based segmentation techniques, some researchers have attempted to combine the strengths of each representation with techniques that make simultaneous use of both types of structure.

Some use of edge information was made in the region–based systems of Ohlander [147] and Ohta [149]. Both systems use the Sobel edge operator to calculate edges for use in the *busyness matrix*. However, this does not serve to directly integrate the two forms of data, but rather to indicate those textured portions of the image over which the hierarchical region segmentation technique is likely to fail.

An edge–constrained approach to region growing has been proposed by Levine [120] that uses a hierarchical edge representation to constrain the region growing process. As a first step, edges that have been created through a locally adaptive process are reduced through a resolution pyramid similar to the processing cone [81]. Then, starting at the top level of this pyramid, region growing is performed on all pixels that are not marked as edges. The region growing process is initiated at the set of seed points that are furthest from edges in the original image. As processing continues down the cone, pixels that have a labeled ancestor inherit that label, while those with unlabeled ancestors are subjected to a new region growing process. Since the original set of edges is not necessarily connected, this iterative process does not constrain the region growing process absolutely, but it does inhibit leakage by limiting the amount of growth for a given region at each iteration.

In a different approach, Kohler [115] does make direct use of both representations through a relaxation procedure that integrates the results of two different segmentations: a region–based Nagin–Kohler [115] segmentation, and an edge–based segmentation according to the algorithm of Hanson, Riseman, and Glazer [84]. Since the edge–based segmentation provides information about the probability of an edge at any given location, this information can be used to determine the influence of a neighboring pixel in the relaxation procedure. The stronger the probability of an edge, the weaker the influence of the pixel on the other side of the edge. However, the algorithm does not take into account the fact that the significance of an edge varies according to the scene content in any given area of the image. Thus, in non-textured image areas the algorithm appears to maintain fine structure during the relaxation procedure, but in textured areas the algorithm tends to preserve the overfragmentation introduced by the region segmentation algorithm.

Trivedi et. al. [186] emphasize the assertion that gray level homogeneity does not provide enough information for good segmentation when there are textural background characteristics. In an attempt to combine the gray level information with edge–based information, they propose an object identification methodology based on a vector of measurements of gray level co–occurrence. This system performs object detection (i.e. region segmentation) using the vector of co–occurrence values for classification. The operation of this system is not very robust, however, since it requires a great deal of supervision in the determination of window size and specification of the co–occurrence parameters used.

In general, attempts to integrate region and edge/line data have not made full use of all available information. Typically, these systems use only local edge estimates rather than true line information. The notion of edge continuity is not used in any of the techniques, except in the case of Kohler's system. Even here, the information was used in the construction of the edge–segmentation, but not during the integration of the two representations. Other considerations, such as

the use of hypothesized edges or lines, are not addressed at all.

2.4 Knowledge–Based Control of Low and Intermediate–Level Processing

One unifying characteristic of the low and intermediate–level algorithms that have been presented in the preceding sections is that the knowledge that is embodied in the algorithms is strictly procedural and linear. In other words, no choice is made during the execution of the process as to what is to be done; the only choice is whether or not to perform the specified action at any given point.

When algorithmic knowledge is specified in a more general form, it becomes possible to provide alternative forms of processing based upon the characteristics of the data. However, the selection of the most appropriate form of processing requires an *evaluation* of the data. As mentioned in Chapter 1, and as will be seen in the following discussion, evaluation of data in the domain of low–level image processing can be a very difficult task.

A number of approaches have been used in image segmentation systems to make use of this form of flexible control. In some cases, the algorithmic knowledge is expressed within a framework that is related to image characteristics, while in other cases the knowledge is structured around semantic (e.g. object–level) characteristics. In the following discussion, the emphasis is on the way knowledge is used in the control of low and intermediate–level processing, rather than on the mechanisms used for semantic interpretation.

Within the general category of knowledge–based segmentation systems, three general classes of approaches are apparent: data–directed semantic grouping systems, top-down goal–directed systems, and integrated systems in which high and low–level processes interact in both a top–down and a bottom–up fashion. Obviously, these distinctions are not clear cut, but it is possible to use this categorization to provide a framework for discussion. The important factors to consider in

an analysis of these systems is the form of knowledge that is being represented and the way in which this knowledge is used to control the behavior of the system.

2.4.1 Data–Directed Semantic Grouping Systems

Data–directed semantic grouping systems are those in which semantic hypotheses related to intermediate–level tokens[3] are used to select and guide those processes which modify the tokens. Much of the early motivation for this approach was derived from research into the three–dimensional interpretation of line drawings [39,42,58,76,104,177,190]. For the most part, these systems ignored the difficult issue of segmentation, and attempted to form three–dimensional models of blocks world scenes with the assumption that the line drawing represented perfect segmentations. Methods for semantic labeling of lines and junctions were developed [58,76,104], and constraint propagation methods were devised to resolve uncertainty in these labelings [104,177,190]. The use of knowledge in such systems (at least with respect to evaluation of tokens) was limited. Since the set of lines was predefined, there was no mechanism for the redefinition of tokens. The use of semantic constraint propagation, however, suggested an equivalent approach for domains in which the set of tokens were dynamically constructed.

Yakimovsky [203] developed one of the first systems that utilized a knowledge–based approach to segment natural scenes. In this system, the segmentation is produced through a two–stage process: a data–directed segmentation based on pixel homogeneity, and a semantically-directed region merging phase. The data–directed initial segmentation process is initiated by using a set of sample points from the image to form seeds for a region growing process. A data–directed region merging technique that evaluates the desirability of a merge according to the boundary strength between regions is then used to complete the data–directed

[3]In order to provide clarity in the description of these various knowledge–based systems, we shall use the term *intermediate–level* to refer to processing which occurs over a set of *tokens*, even though this terminology may not have been used in the original work.

phase of processing. Following a semantic labeling process in which each of the regions is assigned a set of potential object labels, a semantically–directed region merge process is used to compute a Bayesian estimate of merge strength for all pairs of adjacent region tokens. This estimate is based not only on the boundary contrast, but also on the set of potential semantic labels for the two region tokens on either side of the boundary.

Knowledge representation in this system is represented both in the set of semantic labeling rules and in the implicit constraint that adjacent region tokens with identical semantic labels are part of the same object. This implicit knowledge about semantic similarity is used explicitly to refine the region merge scores, and thereby aid in the control of the intermediate–level merge process. The characteristics of the regions themselves are not evaluated in this process; rather it is the characteristics of the adjacency relation between pairs of regions that are evaluated. The processing is strictly bottom–up, since there is no mechanism by which an individual region may be decomposed into smaller regions. Despite these limitations, Yakimovsky's approach has had a major influence on the concept of interaction between high and intermediate levels of processing in image interpretation. Control of the merge process is focused at the intermediate level, both on the basis of intermediate–level characteristics and on the basis of high–level interpretation hypotheses.

This approach is extended in the Interpretation Guided Segmentation (IGS) system of Tenenbaum and Barrow [183]. Although this is also a strictly bottom–up approach (with respect to the initial region boundaries), the interaction between the intermediate and high levels is somewhat more complex than that of Yakimovsky. Working under the basic assumption that general data–directed, domain independent object segmentation is impossible in principle [61], Tenenbaum and Barrow attempt to segment images using local semantic context as well as local data context. In some sense, this system is similar to the region growing algorithm of Griffith et. al. [18] with the addition of semantic labeling rules and

the loose semantic adjacency constraints.

The initial segmentation for the IGS system is a highly oversegmented labeling in which all identical connected pixels are grouped into regions. These regions are then iteratively merged, with the merging criteria based on data constraints (similarity of region features) as well as semantic spatial constraints. The data constraints make use of the contrast across the mutual boundary of adjacent regions to evaluate the desirability of a merge of the two regions. Semantic merge constraints dictate that two regions may not be merged if they do not have at least one common semantic label.

According to the IGS protocol, all initial regions of the segmentation are given the set of all of the possible object labels; the label sets are then reduced according to a constraint filtering algorithm. After semantic constraint filtering, all regions are evaluated for potential merges and those region pairs for which the score exceeds a threshold are merged. This entire process is then iteratively repeated until all possible merges have been performed.

With respect to the concept of knowledge–based control, the impact of the IGS system is that it introduces a new level of interaction between high–level constraint filtering and intermediate–level region merging. The knowledge of semantic identity that is used to modify the intermediate–level region merge process becomes dynamic, and the high and intermediate–level processes of this system cooperate in the refinement of the intermediate–level region tokens.

In a further refinement of this approach, Levine [120,122] makes a conceptual distinction between the tokens created by strictly low–level process and those from processes which utilize semantic knowledge; these are termed *partial* and *complete* segmentations respectively. A partial segmentation is the set of region tokens derived from a low–level processes, where no assumption is made about any correspondence between region boundaries and object boundaries. A complete segmentation is the set of tokens created from the partial segmentation through the use of semantic constraints. However, despite this integrated view of

interpretation processing, the basic control structure for Levine's system is constructed in a linear bottom–up fashion. The tokens of the partial segmentation are produced through a region growing process that uses only local pixel context. Semantic hypotheses are assigned to these tokens via a template matching procedure, and grouping processes are used to merge adjacent tokens into the semantically uniform tokens of the complete segmentation. Although this system is capable of producing highly accurate interpretations of natural scenes [122], it is important to note that the bottom–up processing, which does not allow region splitting, makes the behavior of the system highly dependent upon the quality of the initial segmentation.

Ohta [148,149] presents a system that represents the next evolutionary step in this line of bottom–up knowledge–based interpretation systems with the introduction of intermediate–level evaluation of tokens for *focus–of–attention.* The initial region segmentation of this system is produced through a "nonpurposive" segmentation algorithm similar to the hierarchical mechanism of Ohlander [147]. The selection of color features in this system is accomplished through the use of the *dynamic K-L transform.* Using a Karhunen-Loeve transform, Ohta attempts to find the most useful color space for segmentation for an individual image, thus introducing dynamic control of image features[4].

The results of the initial region segmentation are stored in a *patchery data structure.* This structure stores the hierarchical structure of the region segmentation as well as spatial relations of regions, lines, boundary segments, vertices, and holes. Significant regions of the segmentation (those with large area) are then identified as *keypatches.* These regions serve as focus–of–attention areas for later processing.

The interpretation of the data in this patchery structure is created with a production rule system. Two types of rules are used. *Scene rules* are those which

[4]However, as will be discussed in Chapter 3, Ohta discovered that the principal components of the feature space were quite constant across a variety of images.

correspond to the global interpretation of the image; these are implemented as rules which assign a semantic label to the keypatches. *Object rules* are activated as a result of the assignment of a label to a keypatch, and attempt to merge neighboring small regions into the keypatch. Matching of rule conditions to the the data uses the mechanism of fuzzy logic [204], and the overall scheduling of the system is controlled through a priority queue that utilizes the degree to which the conditions of rules match the characteristics of the data to which they are to be applied.

It is the keypatch mechanism that allows a form of what we would call intermediate–level control. The knowledge that is used to control intermediate–level processing is expressed in the condition portions of the object rules. The creation of semantic hypotheses according to the high–level scene rules can be viewed as an activation condition for the specialized intermediate–level control mechanisms contained in the object rules.

A different approach to intermediate–level focus of attention and knowledge–based control comes from a system for the interpretation of aerial photographs that has been developed by Nagao et. al. [134,135]. Given an initial segmentation produced through a region growing process, this system evaluates the set of region tokens with respect to five different classes of objects. The set of tokens with strong hypotheses corresponding to one of these classes are called *cue* regions. A merge process is then applied to each of the cue regions, using merge criteria specific to the particular class. The significance of this approach is that high–level hypotheses are being used, not to determine whether or not to merge the regions, but to select which intermediate–level rules will be used in the merge evaluation.

A somewhat similar mechanism is proposed by Shirai [178] using edges as the primitive tokens. Following an initial low–level edge detection process, intermediate–level grouping of lines is performed using characteristics of adjacency, type, and gradient direction as the grouping criteria. These sets of grouped lines, call *edge kernels*, are used by high–level interpretation processes for the assignment of ob-

ject hypotheses. Strong hypotheses for the existence of objects then provide the focus–of–attention mechanisms that redirect the intermediate–level grouping processes in an attempt to find specific edge kernels that can confirm the existence of the object.

A more robust and elegant system, based on the same control paradigm, is the ACRONYM system of Brooks and Binford [27,28,29,30,42]. As a domain–independent, model–driven interpretation system, ACRONYM is designed around four interdependent levels of processing: *modeling, prediction, description,* and *interpretation.* In relation to this discussion, however, we are primarily concerned with *prediction,* which is the mechanism through which the system performs intermediate–control. Prediction is implemented through a data structure known as the *prediction graph*: a graph in which the nodes are predictions for the existence of intermediate–level tokens and the arcs between these nodes indicate predicted relations between these tokens.

Initial segmentation processing for this system is based on a line extraction technique developed by Nevatia and Babu [143]. Intermediate–level processes then group the primitive line tokens into a new type of intermediate–level token known as a *ribbon.* A ribbon is a structure with two–dimensional extent that is created from a collection of parallel or nearly parallel primitive edges. It is this set of intermediate–level ribbon tokens that are matched to the object model descriptions represented in the prediction graph, using a constraint manipulation system.

2.4.2 Top–Down Goal–Directed Segmentation

Top–down goal based segmentation systems are those that utilize static goal knowledge to restrict the tokens of the intermediate level representation to those that can be useful in the satisfaction of a particular goal. Early systems of this type made use of static goals, i.e. goals that were known to the system prior to the initiation of segmentation processing. Rosenthal [171] constructed a

system that works under the assumption that only a limited set of specific objects in the scene need to be identified. Consequently the entire image need not be completely segmented. Object descriptions in this system are expressed in terms of contextual properties (on, adjacent), visual properties (size, shape, reflectance, homogeneity), and the regularity-in-spatial-relations (parallel, sequential, unit). These object descriptions are utilized by the strategy generator component of the system to create regions which correspond to the objects of interest.

The first stage of processing in this system is context generation, a process in which an interactive query system is used to determine the object of interest. Given the object name, the system uses the stored object description to calculate the area of the image in which the object is most likely to occur. Using this focus-of-attention area, the system then uses a pyramid structure to compute the level of the pyramid at which the object will be most easily detectable. Given the predicted area and level of the pyramid, a region growing algorithm is then used to produce a segmentation consisting of regions with homogeneous gray-scale values. This segmentation provides a context within which the strategy generator can operate to refine the segmentation of the object in question.

Ballard [7] and Ballard, Brown and Feldman [8,10] present a different type of goal–directed system with the specific goal of determining rib structure in chest radiograms. The system does not start with an initial segmentation, but rather uses a set of short–term plans that can identify particular rib structures. The default start–up plan for this system is an action that can usually be expected to succeed in the location of a particular rib in the chest x–ray. All other plans of the system are expected to make use of the rib hypothesis generated by this start–up plan in the attempt attempt to locate additional rib structures. Thus, once the initial rib is found, predicted locations for neighboring ribs are generated and used by the top–level executive to invoke the rib–finding procedures in order of expected utility. The rib–finding procedures utilize an intermediate level data structure known as the *sketch map* to form a match between the models of ribs

in the database and the location of edges in the image.

Another example of system that uses specific prespecified goals is the Verification Vision System of Bolles [22], that uses template descriptions of binary images to identify objects. Faugeras and Price [59] take a somewhat similar approach in an attempt to determine graph endomorphisms between image segments (from regions and linear feature algorithms) to identify objects. Somewhat more restricted in domain, but more clearly an example of top–down goal–directed segmentation, is the system in which Tucker [188] uses domain knowledge to control a hierarchical quadtree segmentation algorithm. Using microscopic images of neurons as the domain, he embeds the domain–specific knowledge in a hierarchical network of frames. Various experts control the split and merge of regions of the quadtree based on the semantic identification of the region.

2.4.3 Integrated Top–Down and Bottom–Up Systems

In several of the knowledge–based systems described in the preceding two sections, there is a form of integration where processing is controlled in both a top–down and bottom–up fashion. However, in these systems, the top–down control does not extend all the way from the top level down to the lowest level. In these systems, top–down control involves the creation of intermediate–level tokens from other tokens. None of the systems create new tokens from the raw image data under high–level control.

Reynolds [164] presents early work in this area which shows how (partial) interpretation results may be used to direct actual segmentation processing. In this system, efficient segmentation of high–resolution black and white aerial imagery is obtained by the use of the hierarchical resolution processing cone [81]. Images are first reduced to a manageable size (i.e. from ($4096x4096$) to ($256x256$)) by averaging and then segmented using the Nagin–Kohler algorithm. Preliminary semantic object labelings for the set of regions in the segmentation are obtained using the spectral characteristics of the region tokens. Regions whose labels indi-

cate that they may be of interest (e.g. buildings, runways, etc.) are then used as masks for resegmentation at a higher level of resolution. When a resegmentation of these regions is produced at a higher level of resolution, it becomes possible to more closely identify buildings by using shape attributes of the resegmented regions. Thus, although this system uses a fixed segmentation algorithm operating on a fixed image feature, high–level hypotheses are being used to spatially direct a data–level segmentation process, integrating the high and low levels in the interpretation process.

The SIGMA system of Hwang et. al. [105,106] is an integrated system which, although it places most of the emphasis on object models and hypothesis generation, is able to select and apply a set of low–level processes in an attempt to create intermediate–level tokens which satisfy a specific high–level goal. The goals for this system are specified interactively by the user (e.g. "find all houses in the image"). Based on the particular goal, interpretation processes select an appropriate segmentation algorithm from an ordered decision table and initiate a segmentation process using this algorithm. If the set of tokens returned by the algorithm is non–null, control returns to the high–level interpretation process. Otherwise, additional segmentation algorithms are chosen until some set of tokens can be produced.

While the set of low–level processes that can be controlled by this system are somewhat primitive (e.g. blob finder, ribbon finder, upper threshold, lower threshold), the system does select appropriate low–level processes in response to a specific semantic goal. Based on the characteristics and interpretation of the tokens produced by the segmentation process, additional low-level processes may be invoked by the interpretation process to verify the existence of other hypothesized objects.

The knowledge representation in Hwang's system is semantically organized; relations between specific semantic hypotheses and specific low–level algorithms are a part of the static knowledge base. A quite different approach is taken by

Nazif and Levine [121,140,141] with an intermediate–level system that makes use of knowledge about tokens and their relation to image data.

The system makes use of an explicit data representation and control mechanism that allows the modular and incremental modification of intermediate–level data so that a variety of techniques can be utilized in the segmentation process. Knowledge of the system is made explicit rather than procedural [187] so that various mechanisms may be explored in a general and complete manner. Quantitative evaluation of intermediate–level data and evaluation measures for global image segmentations are used to provide both run–time parameter modification as well as experimental evaluation of the system. Finally, they implement a multi–leveled control mechanism for explicit control of the segmentation process.

The system is primarily region–based, in that the final output is a region segmentation. Lines, however, are used extensively during segmentation processing to guide the region formation process. A short term memory (STM) structure is used to store all intermediate–level data, while a long term memory structure (LTM) stores a model that represents the system's knowledge about low level segmentation techniques and control strategies.

Three forms of intermediate level tokens are represented in the system. The initial set of *region* tokens is created through a simple region growing algorithm, and *lines* are produced using a thresholded gradient operation followed by a grouping process based on gradient direction. A third STM primitive, which has no initial representation but is created during processing, is the *area* structure that represents a specific subarea of the image. *Areas* are used by the system for focus of attention and allow the system to direct processing over a set of regions and lines.

Given this initial representation, the system utilizes a production rule system to refine the segmentation. Rules of the system are expressed as condition–action pairs, and are organized in a three–level hierarchy. The first level of rules encodes information about properties of regions, lines, and areas. The actions for

these rules includes region splitting, region merging, line addition, line deletion, line extension, area creation, and area modification. Based solely on the data characteristics of the intermediate–level tokens, these intermediate–level rules independently determine whether or not to fire; the result of executing a rule is a modification of the intermediate–level representation.

The second level of rules encode control information. These include focus of attention rules that order the processing and the metarules that control the application of various rule sets. As such, they are very similar to the metarules of the TEIRESIAS system [45]. The third and final level of rules for the system consists of a set of strategy rules. These strategy rules act as meta–meta–rules; they select the set of control rules that represent the most effective strategy for processing. Based on the quantitative global segmentation evaluation measures, these rules select the most appropriate direction for processing.

This system provides a powerful demonstration of the value of intermediate–level control of processing. Through the explicit expression of evaluation rules and control strategies, very high quality segmentation data can be produced. What is missing, however, is the concept of goal context. The system would not lend itself well to control through a complex set of (interacting) strategies based on high–level expectations. The system does not provide mechanisms by which alternative processing pathways may be evaluated with respect to the manner in which the resulting data corresponds to various interpretation goals. The deterministic nature of a production rule system does not lend itself well to complex sets of strategies without introducing more and more levels of rules, thereby making the system obscure and losing the advantage of an explicit knowledge representation.

2.5 Conclusion

This chapter has provided a background into the state of the art in low–level image processing. We have demonstrated a wide variety of useful techniques

that have been developed to represent and segment image data, and have examined methods which are used to relate segmentation processing to the semantic interpretation process.

The most general comment which can be made about this collection is that each has been shown to be effective within some domain. Some are more general than others, but there are situations in which each will fail to produce useful data.

Methods for the use of knowledge to conditionally control such techniques have also been explored. What is lacking in these systems is a unified structure in which high, intermediate, and low–level processes can interact in both a top–down and bottom–up fashion.

In the next three chapters, we will describe the low and intermediate–level processes, data structures, and control structures that make it possible to tightly integrate these three levels of processing in GOLDIE.

Chapter 3

Low–Level and Intermediate–Level Processes

The set of low and intermediate–level image processes available to GOLDIE form the set of tools that can be used by control processes to create or modify the intermediate–level abstractions that are expressed as tokens. This chapter provides a description of these tools, including: calculation and management of image features, preprocessing of image data, application of evaluation rules, and processes for region segmentation, line extraction, line segmentation, and grouping of tokens.

This set of tools represents a wide variety of techniques that can be used as part of the overall image interpretation process. Obviously, however, there are many other low–level tools that could also have been integrated into the system. The choice of which tools to include has been made on the basis of proven effectiveness in the domain of natural scenes, and on the ease of implementation in the VISIONS environment. The strength of this system lies not in the particular tools, but rather in the way the entire toolset may be controlled within a goal–directed paradigm.

The discussion of the various processes that serve as these tools will be for the most part at the functional level. The intent is to provide a description complete enough for an understanding of the algorithms without becoming too engrossed in implementational details.

3.1 Image Features

As discussed in Chapter 1, the image data for the low–level processes of the GOLDIE system are represented through the set of image features. Low–level

processes are those which operate over the domain of data expressed through these image features. Region segmentation, line extraction, and feature–based measurement of intermediate–level tokens are all processes that require access to this data. The selection of appropriate image features for these local processes can dramatically affect the quality of the segmentation data which is presented to the interpretation processes.

The set of image features currently defined in GOLDIE contains image transformations such as color space rotation [149,170], as well as other types of features such as extrema density (Section 3.1.2) or local area based texture features [91]. These various image features are designed to make explicit, at a local level, characteristics of the image data that are not necessarily apparent in the original RGB or Intensity data. For example, in the calculation of the Variance image feature, a spatial relation representing the variance of the frequency distribution of pixel values in the local neighborhood is expressed as the value of each pixel of the image feature plane. Through this mechanism, local pixel–based classification processes such as region segmentation can make use of information which has no explicit representation in the raw data.

An image feature is defined as a two dimensional array of data that has been computed from the original image data and which is in one–to–one spatial registration with that image data. Since the image feature data is represented in the same data structures as the original image data, the distinction between these two types of data is transparent to the image operators of the VISIONS system. The arrays for each of the image features are represented as *planes* of the VISIONS system, and may be accessed from all levels of the system by a unique *plane index*. The GOLDIE system currently contains the knowledge necessary for the computation and use of 35 different image features (Table 3.1). An image feature is represented in the intermediate–level data structures of GOLDIE (*cf.* Chapter 4) as a graph node (Figure 3.1) that is labeled with the name of the feature (e.g. "Hue"). The value associated with this node is an attribute list that contains

Table 3.1: Image Features of the GOLDIE System

Blue	*Deviation*	*Energy*
Entropy	*Extremadensity*	*Green*
Hepua	*Hue*	*I*
I2	*I3*	*Intensity*
Itwob	*Itwog*	*Itwor*
Lightness	*Mean*	*Normblue*
Normgreen	*Normred*	*Objfeat*
Objfeat-bush-grass	*Objfeat-bush-sky*	*Objfeat-grass-sky*
Objfeat-grass-tree	*Objfeat-sky-tree*	*Q*
Red	*Saturation*	*Slfeat*
Surfdir	*Surflen*	*Variance*
Vepua	*Y*	

```
Hue
     value= '((compute . fdb-hls-plane)
              (imgtype . rgb)
              (name . Hue)
              (no-enhance-flag . nil)
              (segfeatextr-mask . nil))
```

Figure 3.1: Feature Node

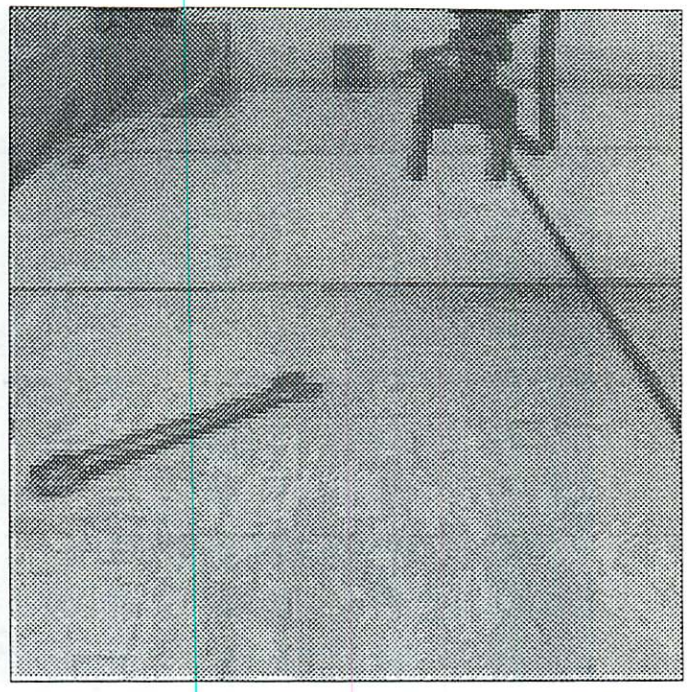


Figure 3.2: Typical Manufacturing Scene

the information necessary for the on–demand calculation of the particular image feature. The actual operation of this on–demand mechanisms will be presented in the discussion of the intermediate–level data structures in Chapter 4.

Input image data for GOLDIE is either a single intensity plane or a set of three RGB planes, depending on the image domain and the type of sensor available. Although the images used in this research (e.g. Figure 1.2) have all been obtained through color (RGB) sensors, the system itself has been made general enough to utilize other types of two–dimensional data, such as obtained from grayscale (Figure 3.2) or tactile sensors (Figure 3.3). With this type of single band data, the system is restricted to operating with image features that can be computed from a single input image.

In the following sections, we will present the image features that are currently used in GOLDIE. Unfortunately, due to limits inherent in this form of publication, color may not be used to provide examples of image feature data, so black

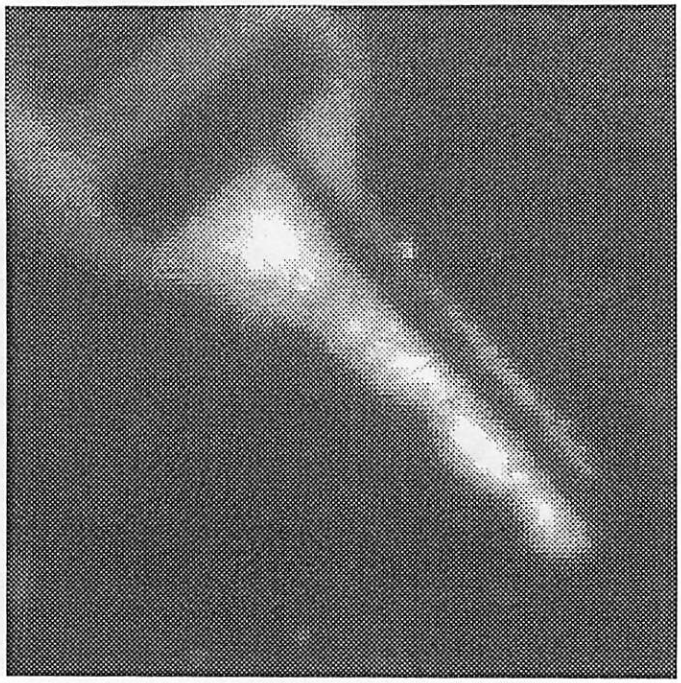


Figure 3.3: Tactile Image of a Key

and white representations must be used to show the features individually. This limitation makes it quite difficult to recognize subtle variations between the image features, so only a small subset of the image features will actually be shown.

For the purpose of the discussion, three general classes of image features may be defined: color image features, textural image features, and semantic image features. Color image features are point transforms in which the feature value at a pixel is a general function of the red, green, and blue values for that pixel. Textural image features encode the spatial distribution of values for a given image feature in the neighborhood of a pixel, and semantic image features represent point transformations of RGB that are determined according to the characteristics of the semantic objects to be distinguished. The definition of the semantic image features is not based on a single static function, but rather on a dynamic function parameterized according to the characteristics of the specific semantic objects involved.

The discussion of these features is somewhat general, and no specific metric is used to compare the utility of image features for segmentation. As was stated in Chapter 1, we have found no generally acceptable method by which such a comparison may be made. As will be shown in Chapters 4 and 6, the specific evaluation of image features can be made only in regard to the set of tokens produced with that feature in a specific image, and with respect to a specific set of goals.

3.1.1 Color Image Features

The color image features include the raw image data and a subset of the image features that are based on mappings of RGB color space that have been shown to be useful in the image interpretation domain (Table 3.2). This set of eighteen image features contains Intensity, RGB, and five other sets of RGB mappings: HLS, YIQ, I1/I2/I3, excess color, and normalized color[1].

The HLS (Hue, Lightness, Saturation) color mapping [62] is a discrete approximation to the IHS (Intensity, Hue, Saturation) model [10], a commonly accepted artistic model of color. In this model, Hue can be viewed as representing the color (average wavelength of light), saturation as the lack of whiteness in the color, and intensity as the total amount of light. Of these three, Hue is especially interesting because of the characteristic that a matte object of constant color, such as a road, should display a constant hue even though part of the road is in direct sunlight and part is in shadow. However, as shown in Figure 3.4, this nice theoretical property of hue is degraded by a combination of factors. Non–uniformity of reflected hue across objects, digitization error, and nonlinearity of sensors all contribute to the somewhat disappointing characteristics of the image. Compounding these difficulties is the problem that hue is an angular measure, and is therefore non–monotonic. Since this is the only image feature in the GOLDIE system that demonstrates this characteristic, each of the low–level processes in

[1]This defines only eighteen features since the I1 of I1/I2/I3 is the same as Intensity

Table 3.2: Color Image Features

Color Image Features	
Feature Name	Description
B(lue)	Original data
G(reen)	Original Data
Hue	Hue from HLS color space mapping [62]
I	I from YIQ color transform [62]
I2	I2 color transform from Ohta [149]
I3	I3 color transform from Ohta
Intensity	$(R+G+B)/3$ or original data
Itwob	$((2*B)-R-G)$ (Blue–Yellow opponent color)
Itwog	$((2*G)-R-B)$ (Green–Magenta opponent color)
Itwor	$((2*R)-G-B)$ (Red–Cyan opponent color)
Lightness	Lightness from HLS color space mapping
Normblue	$B/\max(1,(R+G+B))$ (Normalized blue)
Normgreen	$G/\max(1,(R+G+B))$ (Normalized green)
Normred	$R/\max(1,(R+G+B))$ (Normalized red)
Q	Q from YIQ color transform
R(ed)	Original data
Saturation	Saturation from HLS color space mapping
Y	Y of YIQ color transform

Figure 3.4: Hue Image Feature

the system are required to have a special case to deal with hue. Despite these difficulties, hue has been found to be useful enough to be maintained as an image feature of GOLDIE.

Saturation also presents a difficulty as an image feature in that it is non–linear at low intensity values (i.e. there is a singularity at zero) [109]. Due to the nature of the feature, small perturbations of a single RGB value may cause large shifts of saturation values at low intensities. For example, in Figure 3.5 the shutters of the house, that are black in the actual scene, but which have a slight blue component in the digitized RGB image, demonstrate very high saturation. Since low intensity data is common in this image domain, this image feature is often too unstable to be used for segmentation purposes, but it can be used for classification processes in which the values of Saturation may be discounted over tokens that have low average intensity. The Lightness image feature is very similar to Intensity, and is included in the set of image features only for completeness.

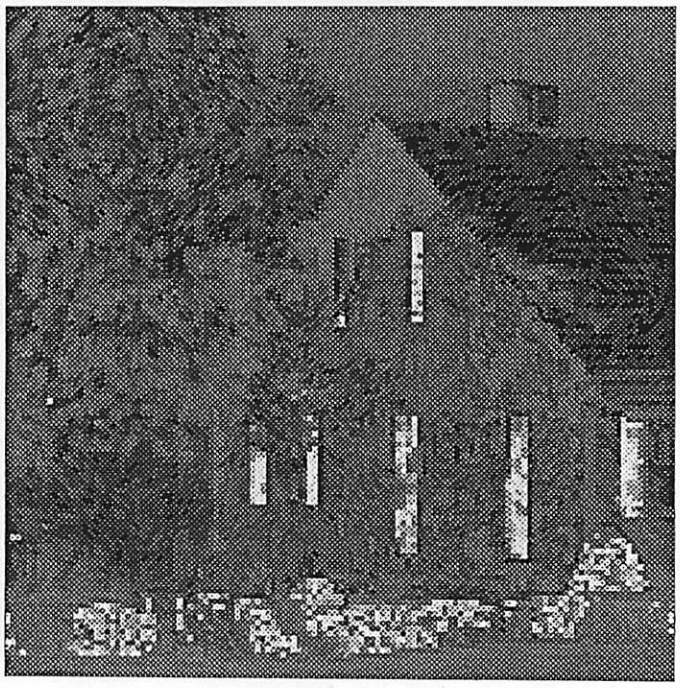


Figure 3.5: Saturation Image Feature

Normalized color (Normred, Normgreen, and Normblue) provide an intensity independent representation of color information (or "chromaticity") that also suffers from the same singularity at low intensity. Additionally, these features suffer from spurious modes and gaps in the frequency distribution due to the discrete quantization of the input RGB data [109] (Figure 3.6). To reduce the effect of this latter problem, these features are scaled in the GOLDIE system through the use of an "anti–aliasing" algorithm that uses the output of a random number generator to minimally smooth the final distribution (Figure 3.7). Typically, these features will be useful in the case that objects of interest in the image are strongly saturated with respect to one of the three original tristimulus RGB values. However, since the intensity information has been removed, this set of features actually contains less information than the original RGB. Therefore, given the unsaturated nature of typical natural scenes, these features are generally less effective than the raw RGB for segmentation purposes, but they can be used to

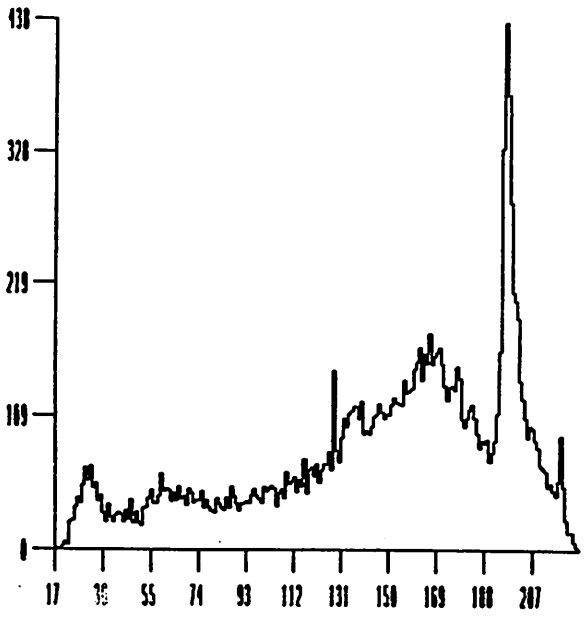


Figure 3.6: Frequency Distribution of Normred Before Scaling

provide intensity independent information for the classification or evaluation of tokens.

The opponent color transformation (Itwor, Itwog, and Itwob) provide a different form of "chromaticity" information that measures, not the ratio of the tristimulus values to intensity, but the "relative excess " of each of these values at each pixel. Although this transform does not suffer from spurious modes or an essential singularity, the intensity information has still been removed. Figures 3.8–3.10 show all three image features produced by this transformation. Despite the fact that frequency distribution of these opponent features typically occupies a relatively small portion of the available range ($2 * Intensity_{max} + 1$), they are often quite useful for segmentation purposes.

The YIQ color transform is the transform specified by the NTSC (National Television Systems Committee) for the transmission of television signals. This linear transform was used by Ohlander [147] as one of a set of three transforms

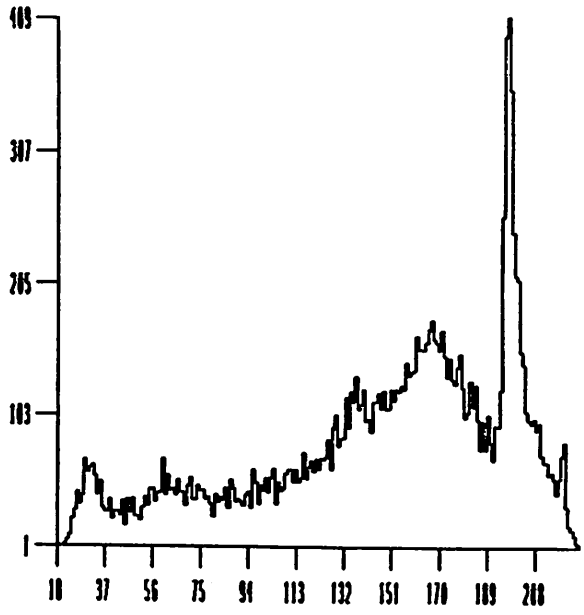


Figure 3.7: Frequency Distribution of Normred After Scaling

utilized by a recursive segmentation system, although he noted that this transform was generally less useful than RGB or IHS. Kender [109], on the other hand, used a rather exhaustive analysis to demonstrate that linear transforms such as YIQ were much more representative of the true image data than the non–linear transforms of IHS and normalized color.

It is noteworthy that the YIQ transform was developed as method to provide compatibility between black–and–white and color television broadcasts, and as such has no explicit foundation for use in image interpretation. Experience in experiments using the GOLDIE system has tended to reinforce the view of Ohlander that this transform does not typically provide the most useful representation, although occasionally the transform will produce images that appear, to an observer, to significantly enhance the differences between different objects in the scene (Figure 3.11).

The I1/I2/I3 color transform was developed by Ohta [149] in an attempt to

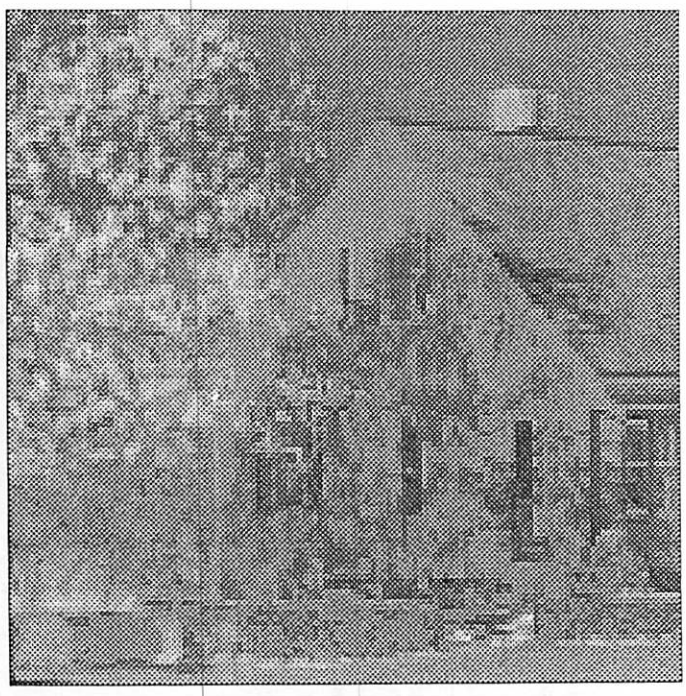


Figure 3.8: Itwor Image Feature

empirically define a set of image features that could maximize the information content of the individual image features for a region segmentation process. Using a database of eight RGB images with varying scene content, Ohta performed a Karhunen–Loeve transform to determine the eigenvectors of the covariance matrix of red, green, and blue values across each of the images. By the nature of the Karhunen–Loeve transform, these eigenvectors must be orthogonal and complete. The principal axis, which corresponds to the largest eigenvalue, is the axis that maximizes variance across the image. The second eigenvector is the axis among those orthogonal to the principal axis that has the largest variance, and the third is defined similarly. In performing, these experiments, Ohta discovered that the eigenvectors selected according to the Karhunen–Loeve transform were very similar across his entire set of images. He therefore selected three fixed features I1, I2, and I3 corresponding to these results.

$$I1 = (Red + Green + Blue)/3$$

Figure 3.9: Itwog Image Feature

$$I2 = (Red - Blue)$$

$$I3 = ((2 * Green) - Red - Blue)/2$$

It is interesting to note that I1 is identical to Intensity and that I3 differs from Itwog only by a scale factor.

The basic assumption used by Ohta in the selection of this feature set is that maximum variance equates to maximum utility for segmentation. However, this is not necessarily the case. Ideally, an image feature will be useful for region segmentation if it has values that are constant across regions and significantly different between regions. Although this type of distribution will typically demonstrate a large variance (assuming a reasonably large number of regions), a large variance does not necessarily mean that the feature will be useful in the segmentation process. For example, a large variance in the Intensity feature across an image containing only trees and bushes does not mean that the resulting segmentation

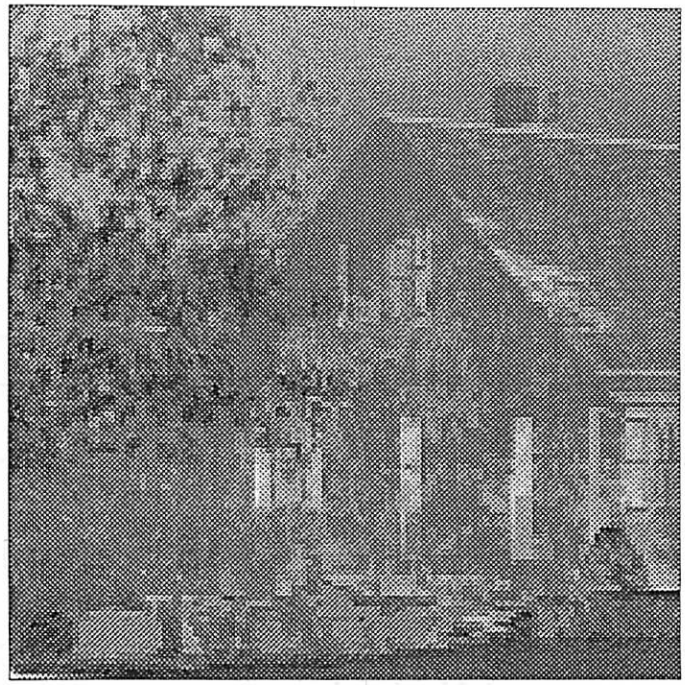


Figure 3.10: Itwob Image Feature

will discriminate these two classes of object; the usual result is that both types of object are quite fragmented. It is for this reason that the selection of features for region segmentation in GOLDIE is conditional upon the desired characteristics of the segmentation to be produced from the image feature. It is our contention that no global measures of feature distribution can produce a reliable estimate of the utility of the feature for the image segmentation process.

3.1.2 Texture Image Features

The set of texture image features is the subset of image features that are typically of use in the segmentation or analysis of textured image areas (Table 3.3). While the color image features are intended to provide a wide variety of views into RGB space, texture image features attempt to provide a view of spatial variation. Feature values of individual pixels represent a measure of the variation of

Table 3.3: Texture Image Features

Texture Image Features	
Feature Name	Description
Deviation	Deviation of Intensity over 5x5 neighborhood.
Edgecontrast	Contrast of the line for support region of which pixel is a member (Section 3.7).
Energy	Energy of frequency distribution of Intensity over 5x5 neighborhood.
Entropy	Entropy of frequency distribution of Intensity over 5x5 neighborhood.
Extremadensity	Gaussian smoothed (3x3) of binary extrema plane in which pixels which have Intensity approximately five percent greater or less than six pixels in 3x3 neighborhood = 1, others = 0.0.
Hepua	Horizontal edge strength.
Mean	Mean of Intensity over 5x5 neighborhood.
Slfeat	Texture feature representing density of short, high contrast lines.
Surfdir	Orientation of the line for support region of which pixel is a member (Section 3.7).
Surflen	Length of the line for support region of which pixel is a member (Section 3.7).
Variance	Variance of Intensity over 5x5 neighborhood.
Vepua	Vertical edge strength.

Figure 3.11: Q Image Feature

light intensity over the local neighborhood of the pixel. Theoretically, the information encoded by these features makes it possible to segment images into areas of homogeneous texture. Of course, a major disadvantage of this approach is the blurring of distinct boundaries in the intensity data due to the spatial extent of the neighborhood mask over which the texture feature is computed. Since texture boundaries (at least in this image domain) are typically somewhat indistinct anyway, the blurring does not present too much difficulty in the placement of these boundaries for regions containing these objects. If the features are used to segment non–textured areas of the image, however, the blurring can produce significant error in the placement of boundaries that are quite distinct in the intensity image. For this reason these features are most effective when segmenting areas of the image that are already known to exhibit textural variation. Segmentation results in Chapters 5 and 6 will demonstrate the effectiveness of this approach.

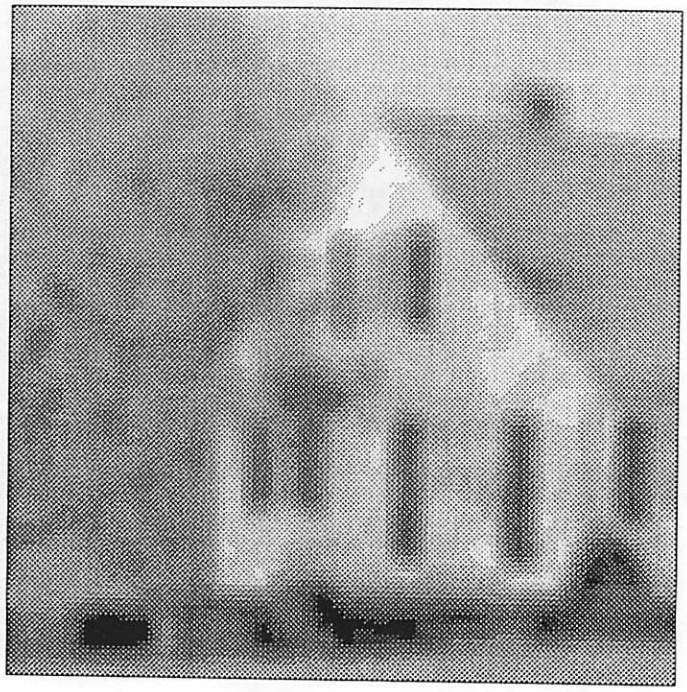


Figure 3.12: Mean Image Feature

The image features Mean (Figure 3.12),

$$Mean(i,j) = \frac{1}{25} \sum_{\substack{i=-2,2 \\ j=-2,2}} Intensity(i,j)$$

Variance (Figure 3.13),

$$Variance(i,j) = \frac{1}{25} \sum_{\substack{i=-2,2 \\ j=-2,2}} (Mean(0,0) - Intensity(i,j))^2$$

and Deviation (Figure 3.14),

$$Deviation(i,j) = \sqrt{Variance(i,j)}$$

represent the respective statistical measures of Intensity values over a $5x5$ neighborhood. Mean is essentially a low–pass filter, and as such does not measure

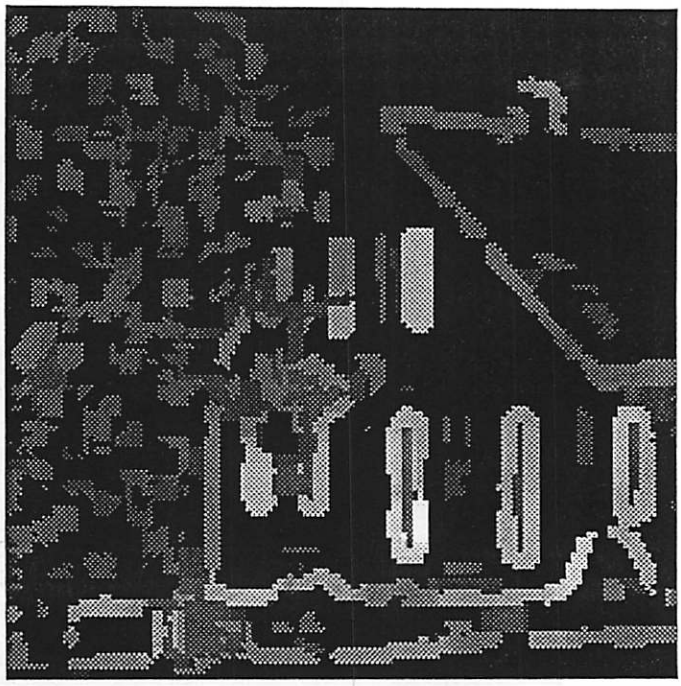


Figure 3.13: Variance Image Feature

texture directly, but can be used effectively to distinguish significant lighting changes (highlights) or color differences in textured areas for which the local high frequency variation would make it difficult for local segmentation algorithms to recognize the more global differences.

Deviation and Variance, on the other hand, are high–pass filters that measure the magnitude of textural variation directly. The use of the $5x5$ neighborhood in the calculation of these features limits the response of the features to slow textural variation with a period greater than 5 pixels, and at the same time minimizes the blurring across true boundaries.

The Extremadensity image feature attempts to measure high frequency texture characteristics through the measurement of the local density of extrema points: pixels which have a value significantly greater or less than most pixels in the local neighborhood. Unfortunately, the feature is highly sensitive to parameter settings that specify the definition of "a significant difference in intensity

Figure 3.14: Deviation Image Feature

values", and we have not found any specific parameter values that are acceptable over a wide range of images. In addition, unless a great deal of Gaussian smoothing is performed (leading to blurring of boundaries) the spatial distribution of this image feature is often quite sparse (Figure 3.15), leading to overfragmentation of region segmentations. However, if very high resolution segmentation of textured areas (i.e. the separation of small areas that do contain extrema points from the area that does not) is desired, this feature can produce acceptable results. Extremadensity can also be useful in object verification when applied to regions.

Energy and Entropy (Figures 3.16 and 3.17) directly measure the amount of organization in the local neighborhood. In the calculation of these two features, the dynamic range of the intensity data is first scaled down from the original

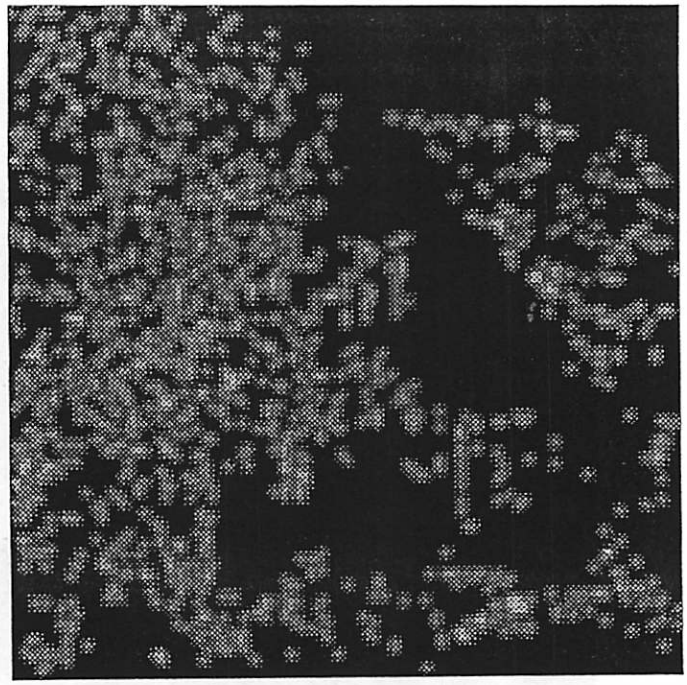


Figure 3.15: Extremadensity Image Feature

$[0:255]$ to $[0:63]^2$, and the the values are calculated as:

$$Energy = \frac{1}{n}\sum_{i=1}^{63} p(i)^2$$

$$Entropy = \frac{1}{n}\sum_{i=1}^{63} p(i)\log p(i)$$

where $p(i)$ is the frequency distribution of scaled intensity values and n is the neighborhood size. These two features provide estimates of the order and disorder in the local neighborhoods, although the small size of the neighborhoods prevents us from considering the values to be statistically valid. Because these features are quite sensitive to homogeneity, they can often be quite useful for region segmentation processes, as will be shown in Chapter 6.

---

[2]This reduction in scale is necessitated by the (relatively) small size of the local neighborhood. The scale reduction effectively reduces the size of the histogram buckets.

Figure 3.16: Energy Image Feature

Hepua and Vepua (horizontal and vertical edge per unit area respectively) are the image features which measure the difference in intensity values for pixels adjacent in the horizontal and vertical directions respectively. These image features are not used in the system for segmentation purposes; instead the mean values of the features are utilized by various evaluation rules that will be described in Section 3.5.2. Surfdir and Surflen are similar in that they too are used solely for evaluation purposes. These image features are based on image data produced during the gradient–orientation based line extraction process that will be described in Section 3.7. Briefly this algorithm is designed to create lines that correspond to image areas exhibiting constant gradient. The Surfdir image feature encodes each pixel with the orientation of the line associated with that pixel while Surflen encodes the length of the line. Measurements of the distributions of values of these two image features across image areas can help in the analysis of textural characteristics For example, Surfdir will be strongly clustered over shingled roof

Figure 3.17: Entropy Image Feature

areas where there is a regular repeating pattern of repeating lines, but will be highly random in trees where there is no regular structure. A final texture feature Slfeat, which is also based on line information, represents the density of short, high contrast lines in the local neighborhood. However, since this image feature is so closely related to the set of lines that are used, the discussion of this feature will be deferred until Section 3.7.

3.1.3 Semantic Image Features

The semantic image features are a special class of color transformations that are designed to utilize semantic information regarding the typical appearance of specific classes of objects to maximize the discrimination of those objects. The set of these features that have been implemented in the GOLDIE system is shown in Table 3.4.

Table 3.4: Semantic Image Features

Semantic Image Features	
Feature Name	Description
Objfeat	Generic color transform feature
Objfeat-bush-grass	Semantic color transform for Bush and Grass
Objfeat-bush-sky	Semantic color transform for Bush and Sky
Objfeat-bush-tree	Semantic color transform for Bush and Tree
Objfeat-grass-sky	Semantic color transform for Grass and Sky
Objfeat-grass-tree	Semantic color transform for Grass and Tree
Objfeat-sky-tree	Semantic color transform for Sky and Tree

The generic function for the calculation of these features involves a linear transformation of RGB space that causes the axis of the new feature to lie parallel to the line between the RGB centroids of the two objects in question. These centroids have been determined by statistical analysis of a large number of labeled segmented images of natural scenes [18], and are stored as values of the object nodes in ILTM. Thus, the Objfeat feature may be calculated for any pair of objects for which this information is known. Since the GOLDIE system currently contains reliable spectral information for tree, grass, bush, and sky, only the six features itemized in Table 3.4 are represented in the system, but the extension to additional object domains involves a trivial extension to ILTM that would encode the spectral information for new classes of objects.

The utility of the semantic image features varies according to the characteristics of the objects in question. If the centroids of the two objects are geometrically close in RGB space, as is the case with Objfeat-grass-tree (Figure 3.18) where:

$$\overline{R}(\text{grass}) = 151, \overline{G}(\text{grass}) = 177, \overline{B}(\text{grass}) = 152$$

and

$$\overline{R}(\text{tree}) = 122, \overline{G}(\text{tree}) = 132, \overline{B}(\text{tree}) = 117$$

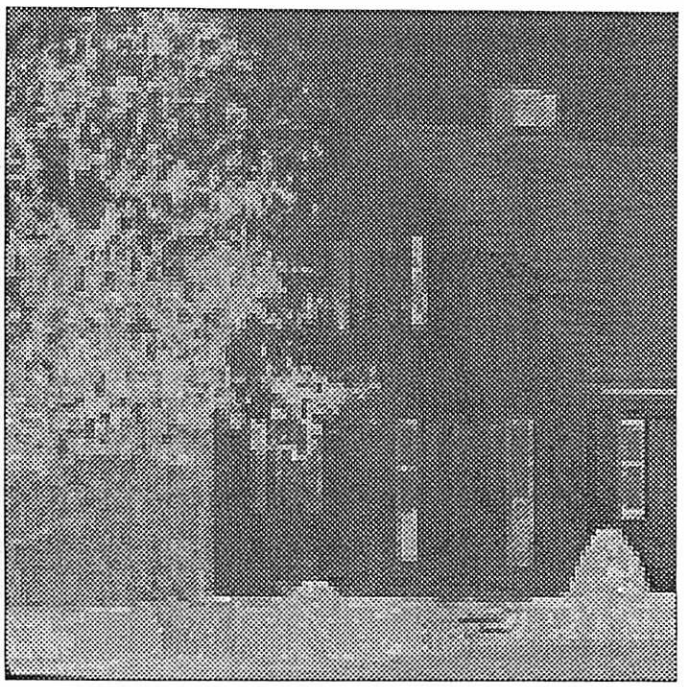


Figure 3.18: Objfeat-grass-tree Image Feature

the discriminant power of the feature is low. However if the axis of the feature is not parallel to the main intensity diagonal of RGB space, and the centroids of the objects are distant in this space, the feature can provide useful information. Figure 3.19 demonstrates the image feature Objfeat-bush-tree which shows a strong distinction between these two type of objects due to the fact that:

$$\overline{R}(\text{bush}) = 59, \overline{G}(\text{bush}) = 90, \overline{B}(\text{bush}) = 46$$

and thus RGB space has been transformed to form this new feature.

3.2 Image Enhancement

Two procedures for image enhancement are available in the GOLDIE system to process image feature data. The selection of which, if any, of these procedures to use is made as a result of a specific goal at system initialization. The rationale

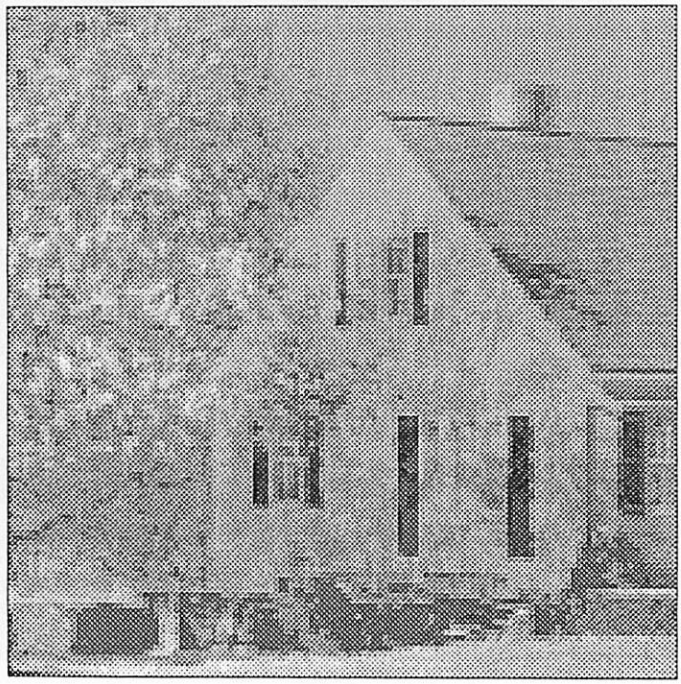


Figure 3.19: Objfeat-bush-tree Image Feature

for this form of global specification is that the construction of knowledge–based processes to investigate the utility of various enhancement procedures for each feature at each stage of the image interpretation process would involve a search space much too large to be computationally feasible. Rather, a selection of one particular enhancement procedure is made initially as a function of the image domain, and the enhancement protocol is consistently observed throughout processing.

Some image feature, such as Extrema, do not lend themselves to image enhancement since the underlying distribution is not continuous. Thus, the system represents this knowledge in the feature node (Figure 3.1) so that the enhancement is not performed on these particular image features when they are computed.

Another potential problem with image enhancement is that the algorithms tend to smooth away textural variation, and would therefore distort the information in the textural image features. Thus, the plane nodes corresponding to each

of the image features maintain two separate pointers to planes representing the raw and enhanced versions of the feature. If the raw version is required, as is the case when Intensity is used to compute the Energy image feature, a special access function is used to retrieve the appropriate data.

The first of the two enhancement algorithms incorporated in GOLDIE is the "edge–preserving" smoothing of Nagao and Matsuyama [133]. In this algorithm pixel values are averaged over a subset of the pixels in the local neighborhood; the subset of pixels is defined such that no pixels that are separated from the central pixel by a high contrast edge are used in the averaging process. In actual implementation, a set of directionally oriented masks are used to select the local neighborhood that has the lowest variance, and the average of pixel values under the selected mask is used as the replacement value for the central pixel.

The second algorithm which has been implemented in this system is the edge enhancement algorithm of Overton and Weymouth [151]. This algorithm employs a weighted average of the values in a fixed size local neighborhood, with the weights being determined as a function of both spatial separation and difference in feature values of the data in the neighborhood. As spatial separation or the difference in intensity values increase, the weights decrease. Thus, the effect is similar to that of the Nagao algorithm in that pixels that are significantly different relative to the variance estimate of the data have little or no effect on the updated value.

In general, neither of these algorithms have been found to be significantly valuable for the operation of GOLDIE in the domain of outdoor natural scenes. In fact, the enhancement processes may occasionally even degrade the quality of the segmentation output. In other domains, however, such as industrial image analysis where the images contain only a few lightly textured objects, these procedures could potentially produce quite beneficial results.

3.3 Region–Based Segmentation

Region–based segmentation processes are used to partition an image or subimage into a number of connected subsets of pixels. To a large degree, these processes form the low–level core of the GOLDIE system in that they create the tokens that are presented to the high–level interpretation processes to serve as the basis for the assignment of semantic labels and structures in the interpretation.

These processes are all controlled through the low–level process controller. To initiate segmentation processing, the controller receives a request that specifies the region of interest, the region segmentation algorithm, the image feature(s), and the postprocessing algorithm to be used. The controller gains access to the required feature data through the intermediate–level data structures, and then invokes the appropriate low–level processes. As these processes complete, they produce a labeled region plane (an image plane in the VISIONS system where each pixel has a unique label identifying the region to which it belongs) that is used by the controller to create the appropriate token data structures in the ISTM.

Five different algorithms for region segmentation have been incorporated in GOLDIE: thresholding, zero–crossings, global one–dimensional histogram clustering, localized one–dimensional histogram clustering, and global two-dimensional histogram clustering. The output of any of these region segmentation algorithms may be postprocessed by either a small region suppression algorithm or by a simple plurality–based relaxation algorithm. Originally, the system also contained an algorithm for full relaxation based postprocessing [169], but as demonstrated by Kohler [115], the computational expense of this process relative to the benefits did not justify its use.

Each of the algorithms has an associated set of parameter sensitivity settings that determine the sensitivity of the algorithm (i.e. the number and type of regions produced by the algorithm). Since there is no generic way of mapping

a single continuous sensitivity value (a value between 0.0 and 1.0) to the wide variety of parameters associated with each of these algorithms, the actual values of the parameters associated with different sensitivities for each algorithm are specified in ILTM. Each of the algorithms has a node in ILTM that contains an ordered list of parameter/value sets associated with different settings. The length of this list determines the number of distinct sensitivity settings associated with the particular algorithm. Thus, if two parameter sets were specified in the list associated with an algorithm, the first parameter set would be used if the sensitivity value were below 0.5, and the second would be used if the sensitivity value exceeded 0.5. This mechanism allows us to determine the range of reasonable sensitivity settings independently for each particular algorithm while maintaining a coherent interface for the low–level process controller.

3.3.1 Thresholding

Segmentation based on thresholding involves the selection of a particular threshold, and the assignment of one label to all pixels that have an image feature value less than the threshold, and a different value to all pixels with values greater than or equal to the threshold. A connected components algorithm is then used to uniquely identify each region of the thresholded image.

In complex images, however, it is generally impossible to find a single threshold that will adequately discriminate between the various objects in the scene. Consequently, other methods must be used in the processing of these images; typically these are recursive thresholding or the selection of multiple thresholds. The recursive approach was employed by Ohlander [147], using a histogram–based threshold selection process that continually applied the thresholding process to all regions in the segmentation until no more thresholds could be selected (effectively a recursive splitting algorithm).

An approach based on the selection of multiple thresholds was proposed by Kohler [114]. Thresholds are selected recursively in such a manner as to maxi-

mize the image contrast across the boundaries of the resulting segmented image. Specifically, given two adjacent (horizontally or vertically) pixels a and b with corresponding feature values $I[a]$ and $I[b]$ we can define the value $N(T)$ as the number of edges detected by threshold T. Assuming (with no loss of generality) $I[a] \leq I[b]$:

$$N(T) = \sum_{alledges} f(a,b,T)$$

where

$$\begin{aligned} f(a,b,T) &= 1 \quad \text{if } I[a] \leq T < I[b] \\ &= 0 \quad \text{otherwise.} \end{aligned}$$

Additionally, if the contrast of the edge between pixels a and b is defined as $c(a,b) = |I[a] - I[b]|$ then $C(T)$, the total contrast of edges detected by threshold T, is defined as:

$$C(T) = \sum_{alledges} g(a,b,T)$$

where

$$\begin{aligned} g(a,b,T) &= c(a,b) \quad \text{if } I[a] \leq T < I[b] \\ &= 0 \qquad\quad \text{otherwise.} \end{aligned}$$

Thus, the average contrast of all edges detected by threshold T would be $C(T)/N(T)$ (assuming $N(T) \neq 0$). A one–pass algorithm is used to compute the average edge contrast for all possible thresholds, and the maximum is selected.

A slight difficulty is that the contrast measure will be biased in cases where a few pixels have values either much greater or much less than all others in the image. This difficulty is easily corrected by setting the average contrast to $C(T)/max(A, N(T))$ where A is a variable representing the minimum number of edges desired.

The algorithm is used recursively by eliminating all edges detected by a previously selected threshold from the calculation of $C(T)$ and $N(T)$. Figures 3.20–3.22 demonstrate the behavior of this algorithm on the Intensity feature of the image

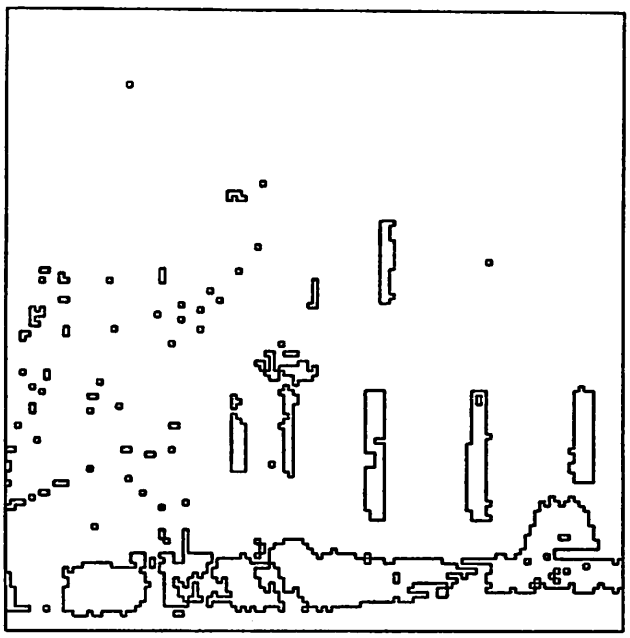


Figure 3.20: Threshold Segmentation - One Threshold

from Figure 1.2 for the thresholds of (70), (70,95), and (70,95,204), respectively.

As can be seen in these figures, the algorithm works well in the selection of thresholds to find many of the significant edges in the image, although some fragmentation of textured areas is inevitable. When used by the schemas of GOLDIE, this algorithm is quite effective in relatively small, non-textured areas of the image where significant boundaries are expected to occur. If the area to be segmented contains a relatively small number of boundary points (e.g. the segmentation of a telephone wire out of a large sky area) this algorithm would lead to a large amount of fragmentation before the correct threshold was located. If, however, the goal is to discriminate shutter from house wall, the algorithm can be quite effective.

The algorithm has been implemented in the GOLDIE system to recursively select thresholds until either a maximum number of thresholds have been selected

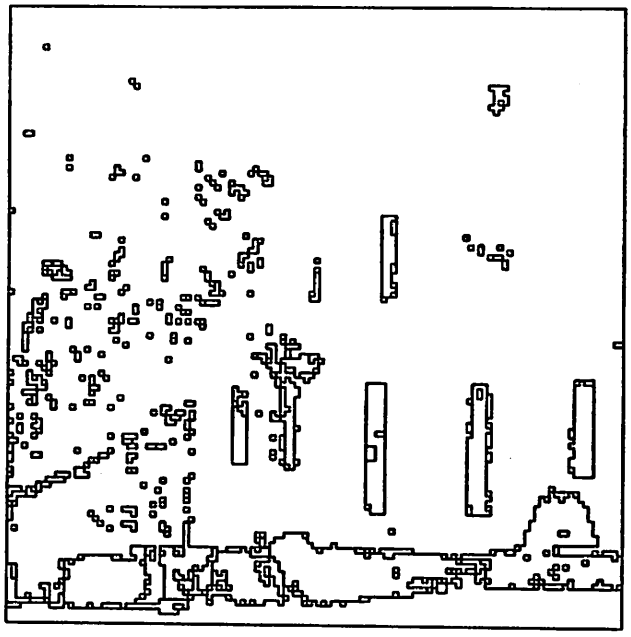


Figure 3.21: Threshold Segmentation - Two Thresholds

or until the average contrast for a selected threshold falls below some minimum value. These two parameters specify the sensitivity for the algorithm, and five different settings based on the parameter/value pairs are specified in ILTM.

3.3.2 Zero–Crossings

The zero-crossing algorithm for region segmentation, initially proposed by Marr [125], is based on the biophysical nature of human perception. By modeling on–center/off-surround receptive fields of retinal ganglion cells, Marr and Hildreth developed the $\nabla^2 G$ operator which can be approximated by the *difference of Gaussians* (DOG). When convolved over the image, this operator computes the non–directional second derivative of the intensity surface of the image. By thresholding the convolved image at zero, a set of boundaries are created, corresponding to the zero crossing edges of the image.

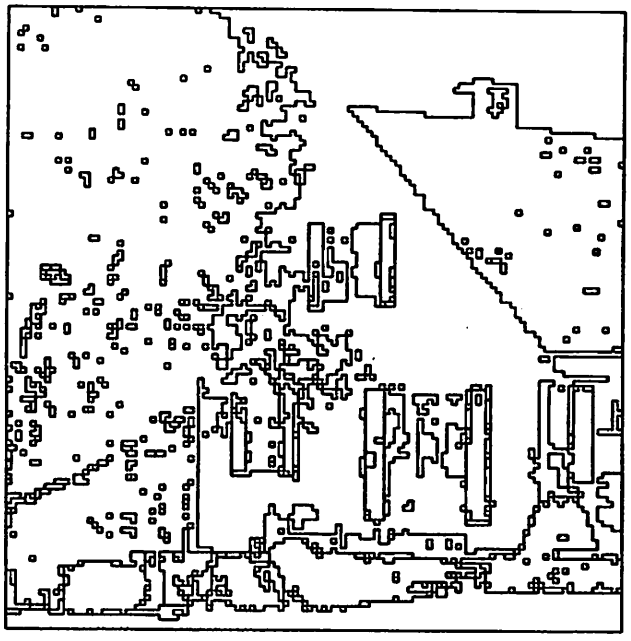


Figure 3.22: Threshold Segmentation - Three Thresholds

Although Marr's approach was to use these edges as the basis of the *primal sketch*, the approach in GOLDIE is to use these boundaries to define a region segmentation. By performing a connected components labeling on the thresholded convolved data, the image is partitioned into a set of closed regions whose boundaries represent significant intensity discontinuities in the image.

Marr also demonstrated the fashion in which the use of different spatial sizes of the DOG operator corresponded to the different *spatial channels* of the human perceptual system. By varying the size of the operator, various levels of detail in the image become apparent. This mechanism has been used to incorporate the various levels of sensitivity for the zero–crossing segmentation operator in the GOLDIE system. Seven different settings are defined, corresponding to decreasing sizes of the Gaussian convolution (i.e. Gaussian radii of 16, 13, 11, 7, 3, 2, and 1 pixels).

Unfortunately, since these boundaries represent the zero–crossing of the Gaus-

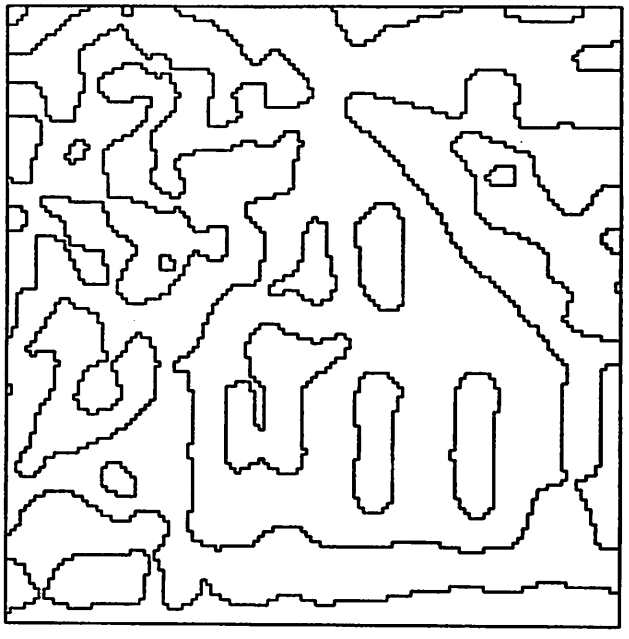


Figure 3.23: Zero–crossing Segmentation - Low Sensitivity

sian smoothed data, the spatial placement of the region boundaries does not always correspond precisely to the intensity discontinuities in the image feature data. This effect is most significant with large operators (low sensitivity settings). Figures 3.23–3.25 demonstrate the segmentation output of this operator on the Intensity feature at sensitivities 0.2, 0.5, and 0.8 respectively (Gaussian radii of 13, 7, and 2).

Another difficulty with this algorithm is that it tends to behave poorly in large areas of slowly changing texture. Although it can effectively partition small textured areas, it can produce very convoluted region structures when applied to large textured areas (e.g. one large region with a large number of holes). For this reason, the algorithm is typically used by GOLDIE only when the goal is to segment a relatively small textured area of the image.

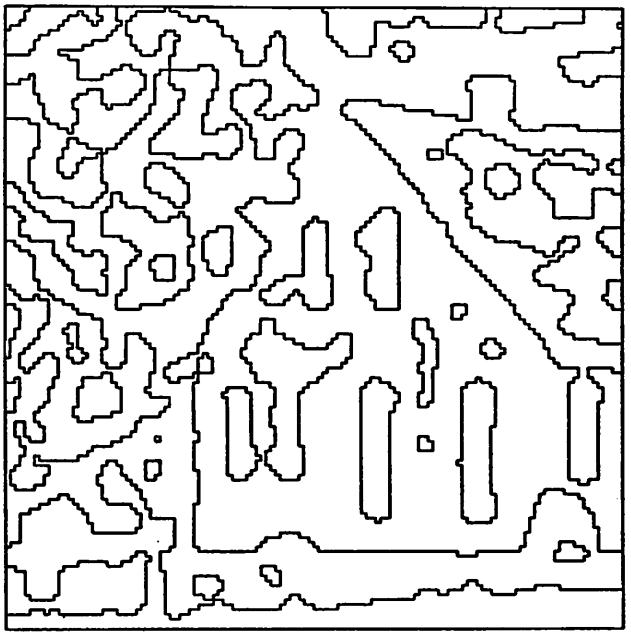


Figure 3.24: Zero–crossing Segmentation - Medium Sensitivity

3.3.3 One–Dimensional Histogram Clustering

One–dimensional histogram clustering algorithms have proven to be among the most durable algorithms for region segmentation. Ohlander [147], Price [160], Nagin [136,137], and Kohler [115] have all demonstrated impressive results through the use of this class of algorithm. The underlying motivation for this approach is that objects in the scene (represented by regions of the segmentation) should produce distinct clusters in the frequency distribution of the image feature (a one–dimensional feature space).

If the objects (regions) to be identified are visually distinct, non–textured, and reasonably uniform in values of the selected image feature, this assumption will generally hold. However, the analysis of a one–dimensional histogram of some feature derived from a reasonably complex scene can be complicated by a number of factors. The most significant of these is the overlapping of clusters. Since the

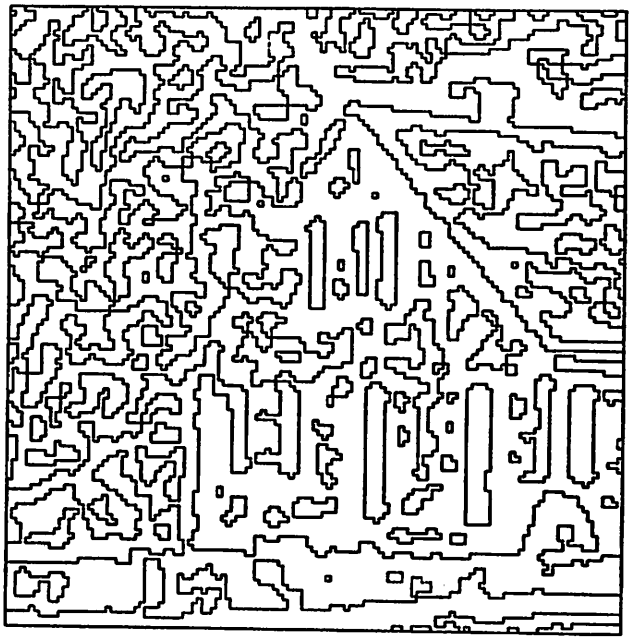


Figure 3.25: Zero–crossing Segmentation - High Sensitivity

clusters typically display a Gaussian distribution, there is often a great deal of overlap between clusters in the histogram that makes it difficult to identify the individual clusters. Another difficulty is that a large, wide cluster can overlap and hide a smaller cluster that is related to a spatially distinct image area. These difficulties can be managed to some degree by the spatial decomposition of the image that is utilized by GOLDIE, and to an even finer degree by the localized peak selection algorithm of Nagin and Kohler.

3.3.3.1 One–Dimensional Global Histogram Clustering

The one–dimensional global histogram clustering algorithm for region segmentation involves the identification and labeling of peaks in the histogram of a given image feature over the area of interest, and the mapping of cluster labels to image pixels. A connected components algorithm is then used to identify the distinct

regions formed by the spatial distribution of cluster labels.

The peak selection algorithm starts with a single pass convolution of the histogram $H(i)$ to perform minimal smoothing, and then identifies all local maxima and minima in the distribution to provide a candidate set of clusters. The candidate set is then pruned according to a three pass parameterized algorithm. Initially, all local maxima with a value less than a minimum height[3] are removed. Next, the peaks (P_i) are sorted in terms of decreasing height, and starting with the largest peak, all other peaks within a minimum distance (in terms of difference in feature values) of an existing peak are deleted. The final criteria is the peak/valley ratio. A valley V_i is defines as the feature value between adjacent peaks P_i and P_{i+1} for which the histogram has a minimum value. Thus, in the peak/valley ratio pruning step, where $H(V_i)$ is the height of the valley at location i, any peak for which:

$$\left[\frac{H(P_i)}{max(H(V_{i-1}), H(V_i))}\right] < PVRATIO$$

is deleted ($PVRATIO$ is one of the algorithm parameters; it defined the minimally acceptable peak–to–valley ratio).

Once a final set of peaks has been identified according to this procedure, the pixels of the image are each assigned the label of the cluster to which it belongs (i.e. all pixels with feature values greater than or equal to V_i and less than V_{i+1} are assigned label i).

The sensitivity settings for this algorithm utilize the minimum peak height, interpeak distance, and peak/valley ratio. Five different settings are defined in ILTM, and Figures 3.26–3.28 show sample segmentations on the Intensity feature for sensitivities 0.2, 0.5, and 0.8 respectively. In general, global one–dimensional histogram clustering is the most robust region segmentation algorithm currently

[3]The parameter value for the minimum height of acceptable clusters is expressed as a percentage of the number of pixels.

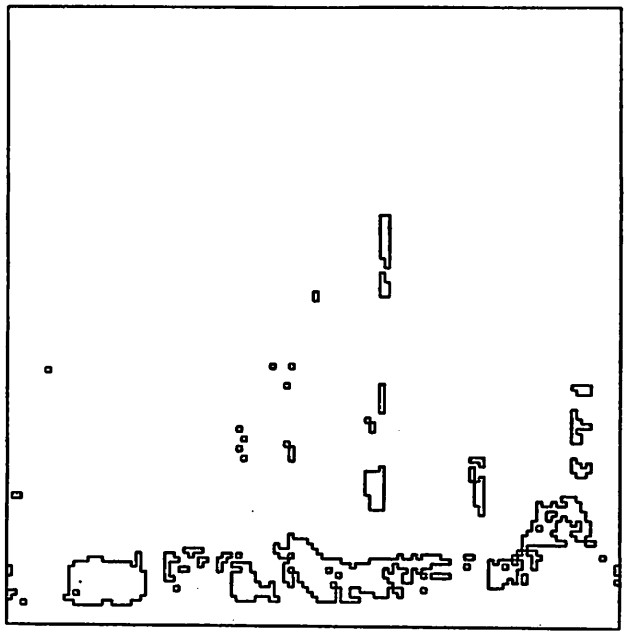


Figure 3.26: Global One–Dimensional Segmentation - Low Sensitivity

implemented in GOLDIE, and provides the best cost/benefit ratio. Thus this algorithm is the one that will typically be used unless specific goal constraints indicate that one of the other algorithms would potentially be more effective.

3.3.3.2 One–Dimensional Localized Histogram Clustering

The enhancement to the global histogram clustering algorithm that was developed by Nagin [136,137] and later refined by Kohler [115] was to partition the image into a set of square subregions (sectors) that could be segmented independently (theoretically in parallel); a complete description of this algorithm may be found in [18]. This procedure provides much greater localization of the histogram in each sector, and therefore better identification of clusters in the distribution. However, since the sector boundaries are arbitrarily determined, several additional procedures are required to remove the artificial segmentation boundaries

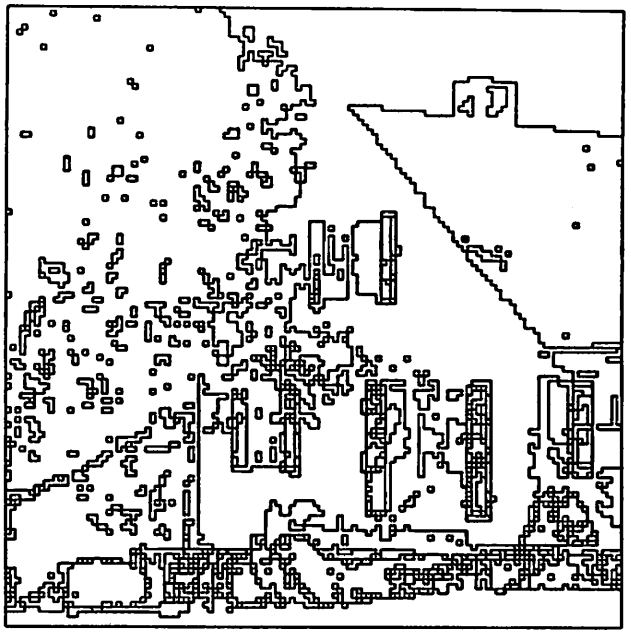


Figure 3.27: Global One–Dimensional Segmentation - Medium Sensitivity

introduced at the sector boundary locations.

Although the recursive approach of the GOLDIE system does provide much of this histogram localization, there are situations (especially with large regions that may be subdivided into several sectors) where the localized peak selection algorithm may be used effectively. Thus, the algorithm is included in the system, and is typically used when very high resolution segmentation output is desired.

There are three areas in which the GOLDIE implementation of this algorithm differs from the global histogram clustering algorithm described above. The first is that the region of interest is subdivided into square sectors of $16x16$ or $32x32$ pixels. To accomplish this, a square mask is set over the region to be segmented, and all data under this mask is used in the segmentation process. However, the set of region tokens actually created by this process is constrained by the area defined by the initial region of interest.

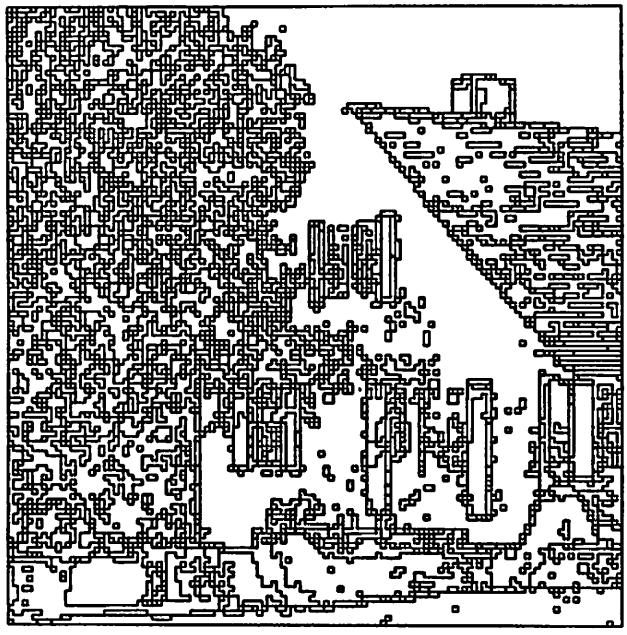


Figure 3.28: Global One–Dimensional Segmentation - High Sensitivity

The second difference between these two algorithms is the "peak propagation" process in which the peaks that have been found in neighboring sectors are considered as potential peaks in the histogram of the current sector [18,115]. This process is motivated by the fact that the sector boundaries are artificially determined, and it is therefore quite plausible that a small portion of a region in a neighboring sector may extend into the current sector. Since only a small number of pixels are involved, the cluster produced by these pixels may not be obvious in the histogram without this peak propagation process.

The final difference between the two segmentation algorithms involves a region merge process which is applied to regions on the sector boundaries [18,115]. Again, since the sector boundaries are artificial, fragmentation may occur in regions which intersect these boundaries. The merge process compares the global mean and variance for the segmentation image feature with respect to each pair of adjacent boundary regions to determine whether or not the regions should be

merged (i.e. the artificial boundaries removed). Although this merge evaluation is not as generalized as the merging algorithms that will be described in Section 3.4, it does remove much of the obvious fragmentation introduced by the sector decomposition.

The primary disadvantage of this algorithm is its computational cost. Since the algorithm uses simulated parallel execution of the segmentation process in each sector, the current implementation uses LISP functions for the intersector communication; consequently the processing time of this algorithm is more than two orders of magnitude greater than that of the global histogram clustering algorithm. Thus the schemas of GOLDIE employ this algorithm over the global one–dimensional histogram clustering only when very high resolution segmentation data is requested. This problem will be eliminated when highly parallel machines, such as the Image Understanding Architecture [191], become generally available.

All of the sensitivity parameters that are used in the global histogram clustering algorithm are also used with this algorithm [18]. The only additional factor that is used to specify the sensitivity settings for this algorithm is the size of the sectors. Five different settings are defined for this algorithm as well, and Figures 3.29–3.31 show sample segmentations on Intensity for sensitivities 0.2, 0.5, and 0.8 respectively.

3.3.4 Two–Dimensional Histogram Clustering

The two–dimensional histogram clustering algorithm of the GOLDIE system makes use of the covariance of two different image features to identify clusters for a segmentation labeling. Since a two–dimensional histogram is itself a two–dimensional array of data, it is possible to use image processing tools to analyze this histogram. Note that in Figure 3.32 the origin (Feature1 = 0, Feature2 = 0) is in the upper left. Feature1 increases to the right, and Feature2 increases to the bottom.

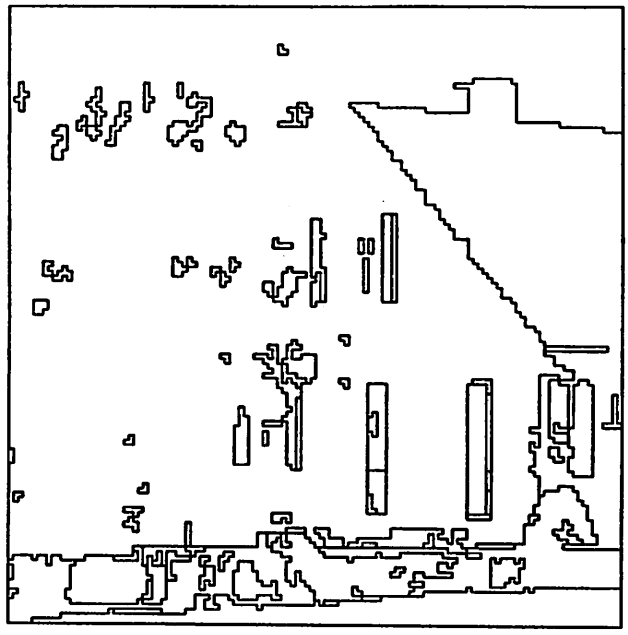


Figure 3.29: Localized One–Dimensional Segmentation - Low Sensitivity

Cluster identification in these histograms is complicated by the fact that correlation between image features tends to collapse the feature space distribution, resulting in ridge–like structures in the histogram that typically run along the main diagonal. This makes it difficult to identify true local maxima and minima. A second difficulty lies in the fact that a cluster in two–dimensional feature space can not be identified by a unique position. The clusters are often elongated and asymmetric; thus the classification metric can not be based on a simple Euclidean distance between a feature point and a cluster center. The difficulties introduced by these two factors make it difficult to label clusters through the simple selection of local maxima. Instead, the approach is to segment the histogram itself with binary thresholding, finding the regions which exhibit the characteristics of local maxima, and then to use these regions as clusters. The two–dimensional clustering algorithm identifies clusters as regions, and the classification metric is based on the Euclidean distance from a point in feature space to the closest point in a

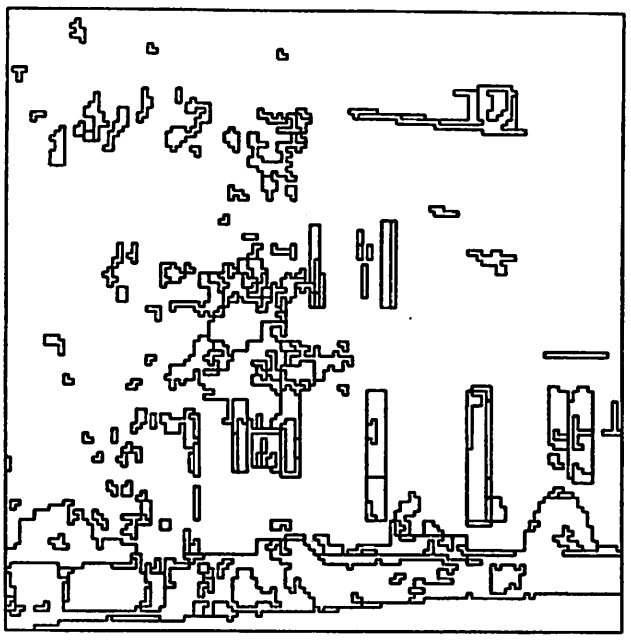


Figure 3.30: Localized One–Dimensional Segmentation - Medium Sensitivity

given cluster.

The actual implementation of the algorithm starts with a Gaussian smoothing of the histogram to remove some local variation. Since the histogram is a sparse matrix, this smoothing helps to reduce the effect of the discrete quantization of the two image features used to construct the histogram. The exact amount of smoothing that takes place is determined by the sensitivity. A dynamic binary thresholding process is then applied to the histogram data in an attempt to find a number of isolated individual peaks. Starting with a threshold value five deviations above the mean value of the histogram, and decreasing by one half deviation at each iteration, thresholds are applied to the histogram until a set of regions are found which can be used as clusters. Acceptability of the set of regions is judged by a parameterized function that measures the number of regions produced and the separation of the regions. Different sensitivity settings are used to modify the evaluation parameters in order to control the number and type of

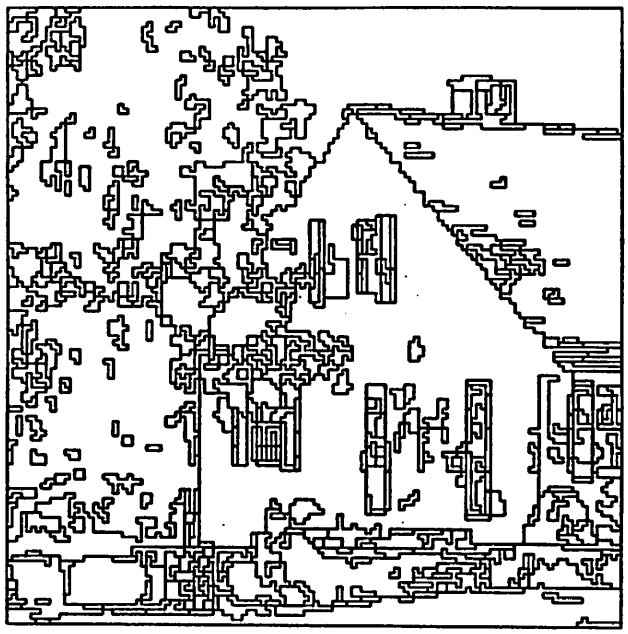


Figure 3.31: Localized One–Dimensional Segmentation - High Sensitivity

peaks selected (see Figures 3.33 and 3.34).

Once an acceptable set of histogram clusters has been selected, a connected components algorithm is used to uniquely label each cluster. Each point not in the histogram regions found by the thresholding process is given the region label of the closest region; this is determined by comparing the Euclidean distance from the point to the nearest pixel in each region and minimizing over this distance (in effect, a minimum distance classifier). The cluster labels are then mapped back to the image, and a connected components analysis is used to compute the segmentation. Figures 3.35–3.37 show the results of this processing at each of the three different sensitivities available (0.2, 0.5, and 0.8 respectively) for the features Intensity and Entropy. As can be seen from these figures, the algorithm is only marginally effective when applied to large areas (in this case, the entire image). Even though one of the two features used for these segmentations was a texture feature (Entropy), the use of Intensity as the other feature broke up the

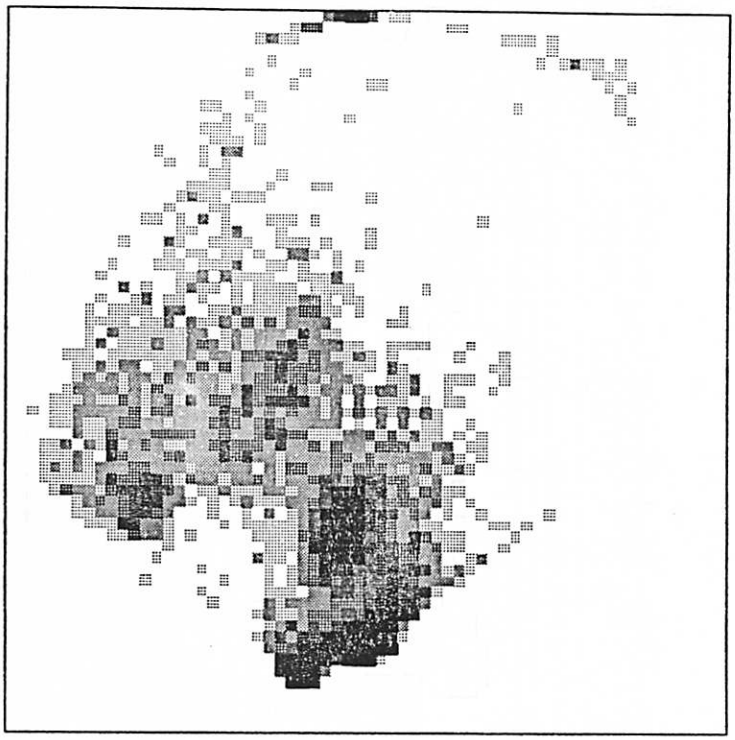


Figure 3.32: Two–Dimensional Histogram of Intensity vs. Itwob

texture clusters and led to a great deal of fragmentation in the tree. However, the algorithm can often be effective in small non-textured areas of the image where several large homogeneous populations can be identified by indistinct clusters in one–dimensional feature space. In these situations, the additional cluster separation provided by the two–dimensional histogram can outweigh the difficulties introduced by the sparse feature space. Thus, the schemas of GOLDIE will often employ this algorithm when the area to be segmented is non-textured, and when the other algorithms of the system have been ineffective.

This algorithm is strongly dependent on the relationship of the two features selected. If the features are very strongly correlated, as shown in Figure 3.38 for Red and Blue, the peaks will all be found on the main diagonal, and the effect will be quite similar to the case when a one–dimensional clustering algorithm were used on Intensity alone. If, however, the features are only loosely correlated, as in Figure 3.32, the algorithm can be quite effective.

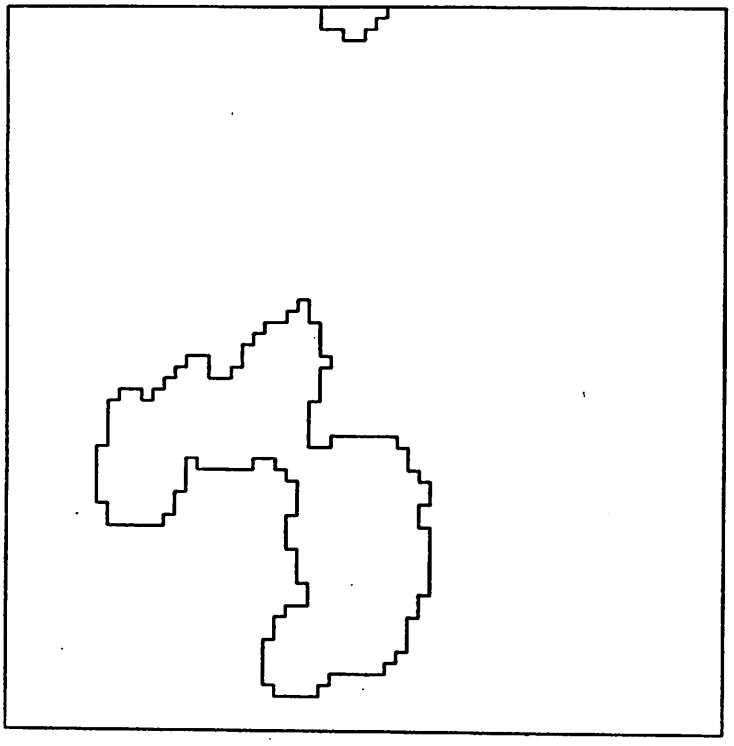


Figure 3.33: Two–Dimensional Histogram Peaks - Low Sensitivity

3.3.5 Segmentation Postprocessing

Once a segmentation labeling has been produced by one of the region segmentation algorithms described in the previous section, two algorithms are available to postprocess the segmentation data in order to reduce the effect of isolated noise in the image feature data. As was seen in many of the figures presented in the previous sections, segmentation output (especially at higher sensitivity settings) often contains a large number of very small regions. In many cases, these regions are the result of local image noise, and are of little interest to an interpretation process. The two postprocessing algorithms are designed to remove these regions in an efficient local manner, avoiding the computational overhead of the more expensive generalized region merging processes.

The selection of a postprocessing algorithm is made as part of the specification of the region segmentation process, but is expressed independently of the

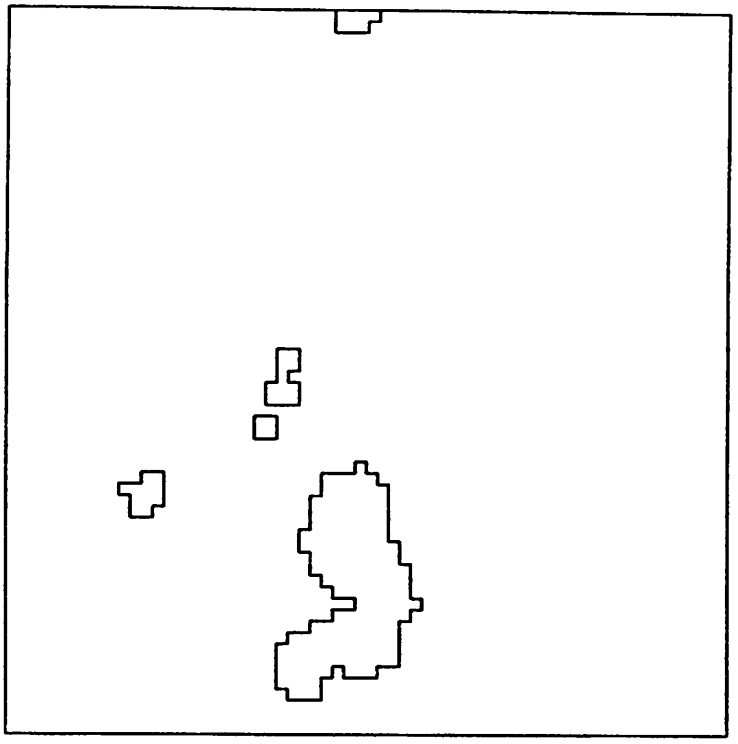


Figure 3.34: Two–Dimensional Histogram Peaks - High Sensitivity

segmentation algorithm. Since postprocessing occurs prior to the posting of region segmentation data into the ISTM, no region tokens are created for such regions.

3.3.5.1 Small Region Suppression

The small region suppression algorithm is a one pass image operator that identifies one or two pixel regions, and merges them into one of the adjacent regions. The selection of which adjacent region to use for the merge may be made on the basis of boundary length, eliminating the need to refer back to image feature data, or it may be based on boundary contrast with respect to a given image feature. Figure 3.39 shows the effect of this algorithm (using the boundary length criterion) on the segmentation from Figure 3.27.

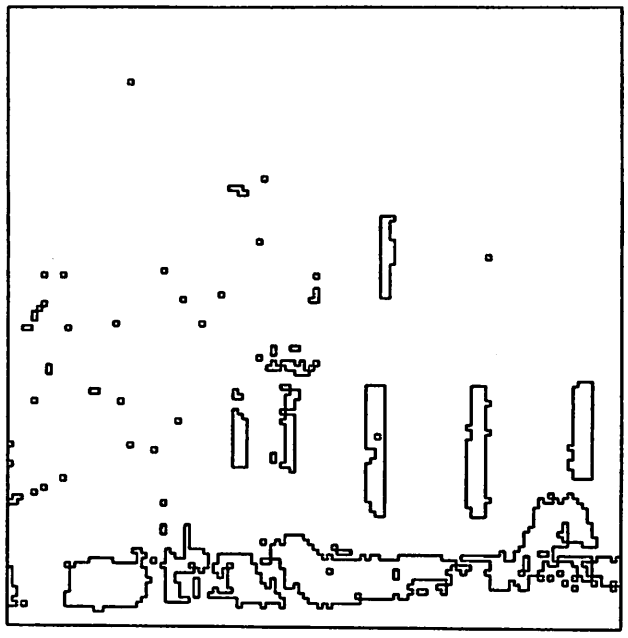


Figure 3.35: Global Two–Dimensional Segmentation - Low Sensitivity

3.3.5.2 Plurality Postprocessing

Although the small region suppression algorithm is quite efficient and effective, there are often cases in which a more general algorithm is desired. For example, a small region consisting of three pixels in a row would not be removed by small region suppression. Given the observation that segmentation data is generally homogeneous (i.e. region boundaries are relatively rare), a relaxation type of postprocessing, in which the cluster labels of neighboring pixels can be used to update the label of the central pixel, becomes quite plausible. Since the small percentage of important small regions can always be regenerated by a later resegmentation processes, there is little disadvantage to removing all fine region structures from the segmentation in the early stages of processing.

The plurality algorithm involves the iterative application of an operator to the cluster label data prior to the application of the connected components process.

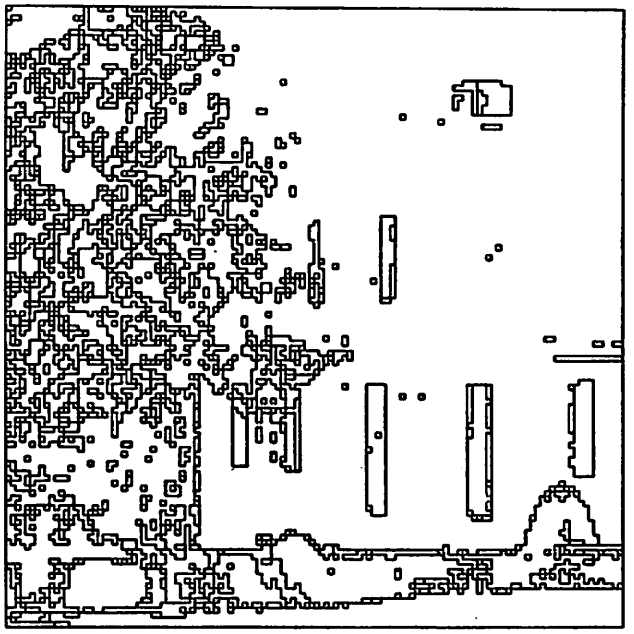


Figure 3.36: Global Two–Dimensional Segmentation - Medium Sensitivity

The image operator is designed to update the label of the central pixel when a majority of the pixels in the 5–*neighborhood* possess a different label. In order to ensure that long single pixel wide regions are not removed by this process, the central pixel label has a weight of two in the voting process. By applying the operator three times in succession, this algorithm can remove regions of up to nine pixels if these regions have a spatial structure that is not compact. In addition to the removal of regions, the algorithm has the effect of smoothing region boundaries, removing single pixel wide intrusions into adjacent regions. Figure 3.40 demonstrates the difference between this algorithm and the small region suppression algorithm.

It is important to note, however, that the smoothing of segmentation data produced by this algorithm is not always desirable. For example, the removal of two small regions near the bottom of the roof in Figure 3.40 has led to the inadvertent merging of the roof and house wall regions. Depending on the goals

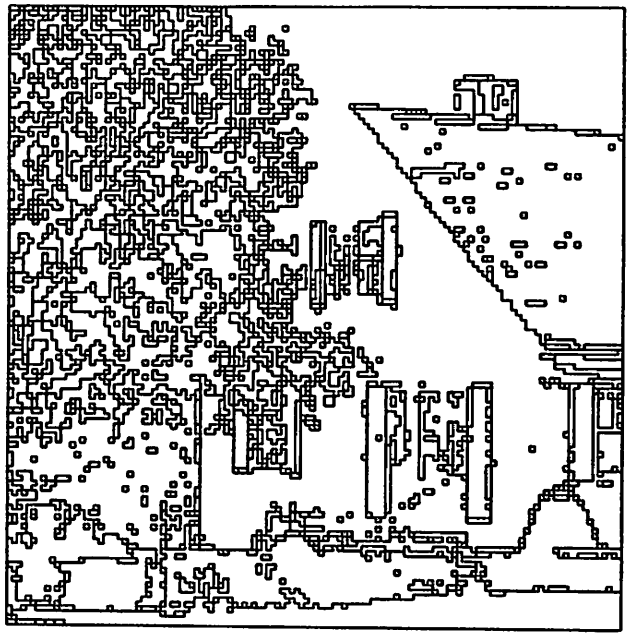


Figure 3.37: Global Two–Dimensional Segmentation - High Sensitivity

of the interpretation process, interpretation may or may not be aided by postprocessing. It is for this reason that the specification of the postprocessing algorithm is made independently of the actual segmentation algorithms.

3.4 Region Merging

The generalized region merging algorithm of GOLDIE involves processes at both the low and intermediate levels of the system. The approach is similar to that of Griffith et. al. [18,49,197], but has been significantly generalized.

3.4.1 Rules For Region Merging

The rules which are used to evaluate a potential merge in the Griffith algorithm may be based on feature values (e.g. the difference of the mean Intensity values for the two regions) or they may be based on spatial relations between the two

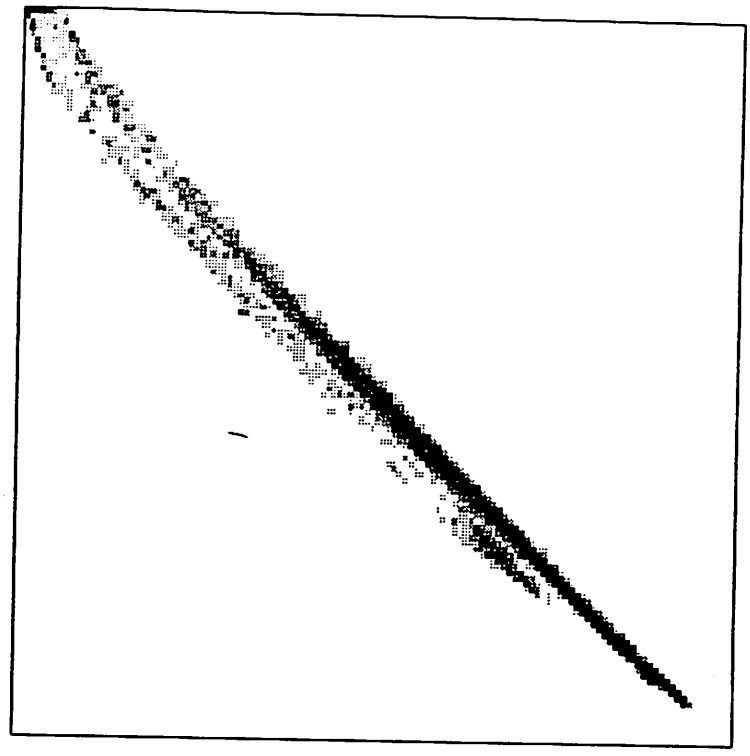


Figure 3.38: Two–Dimensional Histogram of Red vs. Blue

regions (e.g. the length of the shared boundary between the regions). Each of the rules is constructed to return a value centered about 1.0, which indicates that no information is contributed by that rule. Rule values less than 1.0 indicate that the merge should not be performed, while values greater than 1.0 indicate a preference for the merge. This protocol allows a multiplicative combination of rule values so that the final result is a value which indicates the combined evidence of the entire set of rules. In practice, the rules are constructed so that they rarely return a value significantly different from 1.0, since this would represent overwhelming evidence for or against the merge. No single factor has been found that can reliably predict the utility of a potential region merge across a variety of image data. By using an approach that prevents any single rule from dominating the merge evaluation process, the process becomes more generalized and responsive to local image characteristics.

The most significant aspect of this protocol for the combination of rule infor-

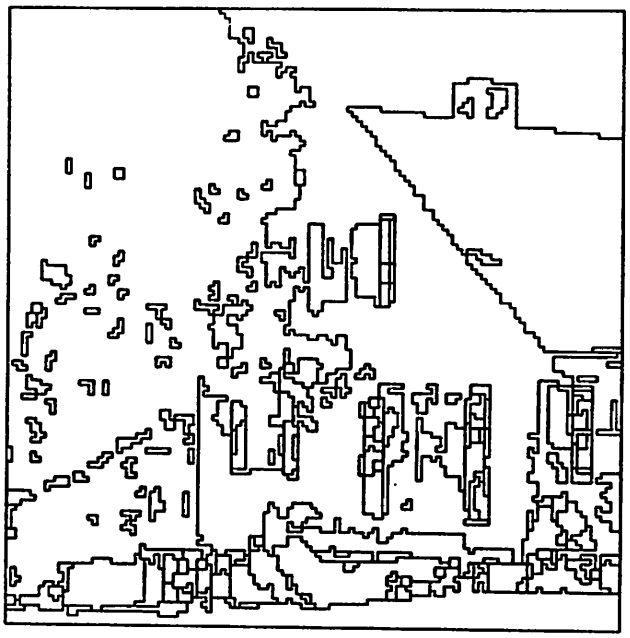


Figure 3.39: Small Region Suppression

mation is that it makes allowance for the errorful nature of rules based strictly on local information. Rather than a strict production rule environment such as that employed by Nazif [121,141] in which the satisfaction of the *if* clause of the rule requires the evaluation of the *then* clause (i.e. perform the merge), the rules of this system provide information as to the desirability of a merge, but do not actually perform the merge. It is the evaluation of the entire set of rules that is important, not the result of a particular rule.

In the system developed by Griffith et. al., a static set of rules was applied (in a simulated parallel fashion) to all pairs of adjacent regions. The most desirable merge (if any) was chosen for each region, and all selected merges were performed. The processing continued until no further merges were selected. A set of seven rules were used, based on:

- Difference of the global means of Intensity normalized by the sum of standard deviations of Intensity for the two regions.

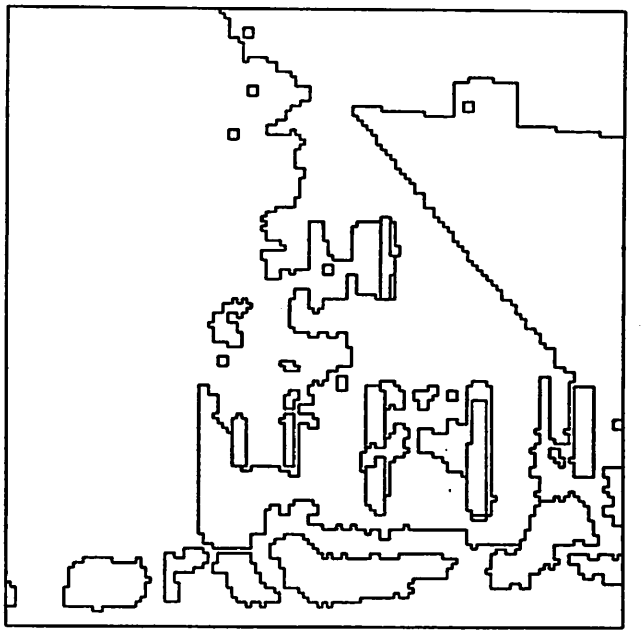


Figure 3.40: Plurality Postprocessing

- Region size - small regions are encouraged to merge with larger regions.
- Degree of adjacency - regions connected by relatively long boundaries are encouraged to merge.
- Similarity of the standard deviations of Intensity.
- Difference of local means of Intensity near the shared boundary.
- Standard deviation of the two regions as a whole.
- The degree to which the region could have been caused by mixed pixels during the digitization process (i.e. a narrow region between regions of locally lower and higher measures).

The nature of this algorithm suggests that it is possible to use dynamic sets of rules to evaluate merge potential, tailoring the rule set to the current goals

of processing. This is the approach to region merge evaluation that is used in GOLDIE. In this model, the evaluation of merge potential between two adjacent regions is performed by a knowledge source using three parameters: the pair of region tokens, a list of rule–feature pairs, and a list of weights corresponding to the rule–feature pairs. Rule–feature pairs express the name of a rule and (if necessary) the name of the image feature which will be used in the evaluation rule. The weights are used to modify the relative contribution of each rule, and are expressed as the exponential power to which the rule result is raised. Weights are defined in the inclusive range [0.0 : 1.0] and therefore can be used to reduce the impact of the rule. Since the weighting factors do not add information, but rather reduce certainty (i.e. bring the rule value closer to the no information state), the weights may be used to allow the use of marginal rules with minimal impact. The merge evaluation rules which are currently used in GOLDIE are summarized in Table 3.5.

Three specific sets of these rules are defined in ILTM as being the default rules to apply in the case that the merge is to be considered in the context of smooth regions, textured regions, or gradient regions. Tables 3.6–3.8 show the rule organization for these different rule sets. Note that the fact that the weight for most rules is 1.0 means that most region merge evaluation rules are fairly reliable. The difference between the various rule sets has to do with the individual rules and image features employed. For example, the smooth region merge evaluation rule set deals with a large number of characteristics that should all be similar if the two regions are to be merged. If the two regions are non–textured they should be quite similar in mean Intensity, planarity, color, and variance of the Intensity feature over the region. Other considerations such as relative size, shape, and boundary contrast can also help to influence the evaluation. For the evaluation of a merge between textured region, on the other hand, some of these characteristics do not represent the underlying nature of the image data. In this case, some of the characteristics, such as planarity, are meaningless, and others,

Table 3.5: Region Merge Evaluation Rules

Rule	Description
ms–boundary	Based on boundary length and region shape. Region pairs with long shared boundary length (relative to total length) receive a high value, long thin regions also receive a high value.
ms–feat–contrast	Based on the boundary contrast between the regions with respect to a specified image feature. The lower the contrast, the higher the value.
ms–feat–mu	Based on the difference of mean values of a specified image feature for the two regions. The smaller the difference, the higher the value.
ms–feat–planefit	Based on the angle between planes fit to the intensity surface of the specified image feature for the two regions. The smaller the angle the higher the value.
ms–feat–sigma	Based on the difference of variance values of a specified image feature for the two regions. Large differences in variance receive a low value.
ms–objhyp–similarity	Based on the similarity of semantic object hypotheses associated with the two regions (returns values ≥ 1.0 only).
ms–rgbmu	Minimum of ms–feat–mu for Red, Green, and Blue.
ms–size	Based on region size. Small regions receive a large value.

Table 3.6: Smooth Region Merge Evaluation Rule Set

Rule	Feature	Weight
ms–feat–mu	Intensity	1.0
ms–feat–sigma	Intensity	1.0
ms–feat–contrast	Intensity	1.0
ms–rgbmu	-	1.0
ms–planefit	-	1.0
ms–size	-	0.5
ms–boundary	-	0.5

Table 3.7: Gradient Region Merge Evaluation Rule Set

Rule	Feature	Weight
ms–feat–contrast	Intensity	1.0
ms–planefit	-	1.0
ms–boundary	-	1.0
ms–size	-	1.0

such as boundary contrast, may give misleading information. Thus the texture merge evaluation rules take into consideration mean Hue and Slfeat, size, and boundary length. By changing the evaluation set to match local image characteristics, the system is able to perform this evaluation in a knowledge directed manner. These three rule sets are based on the specific types of characteristics for intermediate–level evaluation that have currently been implemented in GOLDIE. The discussion of intermediate–level representation and control in Chapters 4 and 5 will demonstrate the way that these particular characterizations are used. For the purposes of this discussion, it is necessary only to recognize that these three specific intermediate–level characterizations represent three particular perspectives on intermediate–level evaluation that are particularly applicable to the domain of outdoor natural scenes.

The fact that these rule sets are defined in ILTM in no way restricts the

Table 3.8: Texture Region Merge Evaluation Rule Set

Rule	Feature	Weight
ms–feat–mu	Hue	1.0
ms–feat–mu	Slfeat	1.0
ms–rgbmu	-	1.0
ms–size	-	1.0
ms–boundary	-	1.0

generality of the rule based approach for region merge evaluation. They have been devised as heuristics which appear to be useful in the domain of natural scene images. Extensions to additional domains would involve the selection of new rule sets which are appropriate to that domain.

This evaluation mechanism appears to be quite robust. As long as rules are constructed conservatively, rarely returning values significantly different from 1.0, the overall behavior is not dramatically affected by any one rule. Experimental results will be presented in Chapter 5 to demonstrate the effectiveness of this rule–based approach.

3.4.2 Region Merge Processing

Once a decision is made to merge a pair of adjacent regions, the actual processing involved in the performance of the merge is quite straightforward. A new region token is created, with a bitmap corresponding to the union of the bitmaps of the merged regions. A link is then created between the new region token and the old tokens, showing that a merge relation exists, and additional entries are made into the intermediate–level data structures that allow the system to perform on–demand feature calculation. Since the feature values of region tokens are expressed through active values, no specific feature measurements need to be stored in the region token. As feature values are requested for this new token, the appropriate measurement processes will be invoked to calculate and store the

desired information.

3.5 Region Evaluation

A key characteristic of GOLDIE is the representation of knowledge that permits the evaluation of region tokens at the intermediate level. This knowledge is represented in a set of rules which may be applied to the region token to produce hypotheses about the nature of the region. The purpose of this evaluation is not to provide an interpretation of the region token, but rather to produce information that can aid in a decision as to whether the existence of the region is supported with respect to both the underlying image data and the current goals of the interpretation process.

Two classes of intermediate–level evaluation are used by the system: *data–oriented evaluation* which forms hypotheses as to whether the region should be resegmented or merged, and *semantic evaluation* which provides information as to the set of possible semantic labels which may be associated with a particular region token. The rules of both classes obey the same conventions as were used in the merge evaluation process; individual rules return a value centered about 1.0, and are multiplicativly combined to produce a hypothesis value.

3.5.1 Data–Oriented Region Evaluation

Data–oriented region evaluation involves the application of a particular subset of the region evaluation rules specified in Table 3.9. These rules are designed to provide direct information about the way in which the image data maps to a region. Two different types of intermediate–level evaluations are performed using various combinations of these rules. The first type of evaluation is designed to measure the degree to which a particular region requires resegmentation. (i.e. the degree to which the region is overmerged). The second form of evaluation is used to measure the degree to which a region should be considered for a merge

Table 3.9: Region Evaluation Rules

Rule	Description
re–adjregs	Based on ratio of size of region to average size of all adjacent regions.
re–compact	Based on "compactness" measure of region extent.
re–dir	Based on variance of orientation of all short lines intersecting the region.
re–extentsratio	Based on the size of the Minimum Bounding Rectangle for the region.
re–feat–contrast	Based on average contrast across region boundary with respect to a specified image feature.
re–feat–contrastratio	Based on ratio of average boundary contrast across the entire perimeter of the region to the average contrast over the lowest contrast region–region boundary with respect to a specified image feature.
re–feat–variance	Based on variance of the specified image feature across the region.
re–feat–planefit	Based on least squares error of a planar fit to the intensity surface of the specified image feature across the region.
re–sld	Based on variance of the Slfeat image feature.
re–mergesize	Based on region size.
re–resegsize	Based on region size.
re–texturedev	Based on variance of local mean line contrast.

Table 3.10: Smooth Region Resegmentation Evaluation Rule Set

Rule	Feature	Weight
re–feat–variance	Intensity	0.5
re–feat–variance	Deviation	1.0
re–resegsize	-	1.0
re–planefit	-	1.0
re–sld	-	1.0
re–extentsratio	-	0.5

Table 3.11: Gradient Region Resegmentation Evaluation Rule Set

Rule	Feature	Weight
re–feat–variance	Deviation	1.0
re–resegsize	-	1.0
re–compact	-	1.0
re–planefit	-	1.0
re–sld	-	1.0
re–extentsratio	-	1.0

with neighboring regions[4]. Regions in a segmentation are not necessarily required to be homogeneous with respect to a particular image feature; rather they are expected to be homogeneous with respect to an interpretation process. Thus the evaluation of a region can not be limited to a simple rule which measures the unimodality [146,149] or variance [140] of an image feature across the region.

In order to clarify this point, Tables 3.10–3.12 specify the rule sets that are stored in ILTM for the creation of hypotheses about the need to resegment a region given the different goal contexts. As was the case with region merge evaluation, these rule sets are organized according to the intermediate–level classifications of *smooth*, *gradient*, and *texture*.

[4]Note that merge potential is a unary measure for a particular region rather than a binary relation between a pair of regions as was the case in Section 3.4.

Table 3.12: Texture Region Resegmentation Evaluation Rule Set

Rule	Feature	Weight
re–feat–variance	Deviation	0.5
re–resegsize	-	1.0
re–texturedev	-	0.5
re–sld	-	1.0
re–extentsratio	-	1.0

If the system were interested in the evaluation of region tokens which were expected to be smooth (with respect to the intensity surface profiles) the fact that the image feature displayed high variance over the region would be a strong indication that there was a need for resegmentation. Thus the rule *re–feat–variance* is applied to the Intensity image feature. This rule returns a low value (indicating little need for resegmentation) if the deviation of the image feature is low, and a high value for high deviation. If textured regions were the desired tokens, however, a much more reasonable rule would measure the variance of a texture image feature (Deviation). Thus, a high value would be returned only in the case that there was significant variation in textural characteristics.

Other factors, such as region size, contribute to the rule set output in the obvious fashion. Large regions are good potential candidates for further segmentation while small regions are not. It is worth emphasizing, however, that the output value of the evaluation is not dependent upon any particular rule value. If a very large region demonstrated low variance in RGB and low short line density, the smooth region resegmentation evaluation rule set would still return a low value.

The particular rules, features, and weights specified in each of these rule sets represent heuristic expectations about the domain that have been refined somewhat through observation of system behavior. It is certainly possible to experimentally derive rule sets, given a large enough training set of segmentations labeled with the characteristics of interest, but it is striking how well the rules

Figure 3.41: Resegmentation Hypotheses with Smooth Constraint

perform in our experiments without such a rigorous procedure. Figures 3.41–3.43 show the sets of regions for which the resegmentation scores are greater than 1.0 using the rule sets for smooth regions, texture regions and gradient regions respectively. With the smooth region rule set (Figure 3.43), nearly all large regions that exhibit variation (e.g. the textured roof region) become candidates for resegmentation. However, when the texture region rule set is used (Figure 3.42), the most important factor is variation in textural characteristics, and thus only those regions that exhibit significant variation in texture become candidates for resegmentation. With the gradient region rule set (Figure 3.43) , regions that exhibit a relatively constant gradient are acceptable, and thus several of the regions near the bottom of the house receive much lower evaluation scores than they did with the smooth region rule set.

The intermediate–level evaluation of whether or not a region token is a likely candidate for a merge process can be adapted to processing goals in a similar fash-

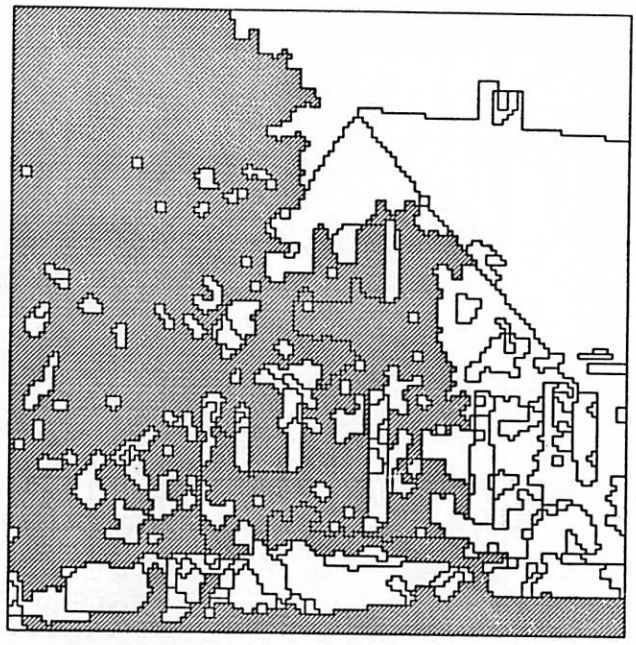


Figure 3.42: Resegmentation Hypotheses with Texture Constraint

Table 3.13: Smooth Region Merge–Potential Evaluation Rule Set

Rule	Feature	Weight
re–mergesize	-	1.0
re–feat–contrast	Intensity	0.5
re–feat–contrastratio	Intensity	0.5

ion (Tables 3.13–3.15). Although the same subset of three rules is used in each of these categories, the weights vary with respect to the desired region characteristics. These rule sets are designed to produce an initial estimate of the desirability of evaluating the set of possible merge scores for a given region. The evaluation of all possible merge scores for a region is expensive, especially if the region is adjacent to a large number of regions. Through the creation of hypotheses about merge potential, it is possible to significantly reduce the number of candidate merge scores that are computed. Figures 3.44–3.46 show the sets of regions that

Figure 3.43: Resegmentation Hypotheses with Gradient Constraint

Table 3.14: Gradient Region Merge–Potential Evaluation Rule Set

Rule	Feature	Weight
re–mergesize	-	1.0
re–feat–contrast	Intensity	0.5
re–feat–contrastratio	Intensity	1.0

Table 3.15: Texture Region Merge–Potential Evaluation Rule Set

Rule	Feature	Weight
re–mergesize	-	1.0
re–feat–contrast	Intensity	0.75
re–feat–contrastratio	Intensity	1.0

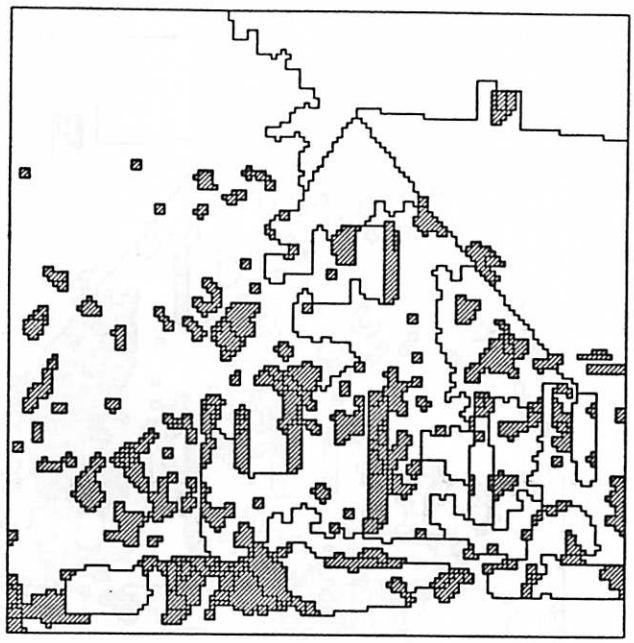


Figure 3.44: Merge–Potential Hypotheses with Smooth Constraint

have merge evaluation scores greater than 1.0 using the merge–potential rule sets for smooth regions, textured regions and gradient regions respectively. Although these three figures are basically quite similar, in that most small regions have high merge evaluation scores, there are a few differences. For example, the scores produced by the texture rule set are higher in areas where there is higher boundary contrast (e.g. boundaries of the house wall). On the other hand, the gradient rule set produces higher scores in the area of the shutters, where average boundary contrast is high, but where, in addition, there are adjacent regions that share a short, low contrast boundary.

With these types of intermediate–level evaluation, GOLDIE gains the capability to use intermediate–level knowledge about the relation of region tokens to the image data, and is therefore able to tailor the intermediate–level evaluation of region tokens to the goals of the system. The addition of different evaluation contexts corresponding to new goal domains simply involves the specification of a

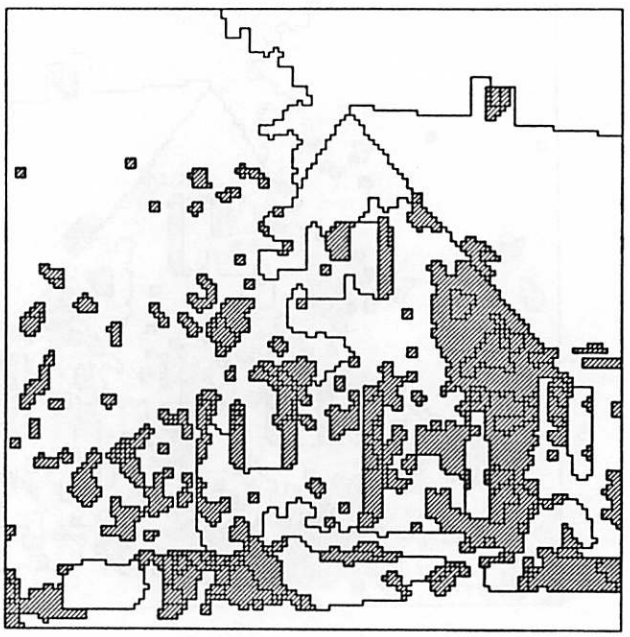


Figure 3.45: Merge–Potential Hypotheses with Texture Constraint

new rule set in ILTM. The discussion of the initialization schema in Chapter 5 will show how this capability can enhance the control of intermediate and low–level processing.

3.5.2 Semantic Region Evaluation

Specific hypotheses about the set of potential semantic labels that may be assigned to a particular region can often be of great help in intermediate–level specification of processes. For example, if a region to be resegmented were known to have strong semantic hypotheses for sky and tree, this information could be used to select the image feature Objfeat-sky-tree for the resegmentation process. Although the creation of semantic hypotheses is properly a function of high–level interpretation processes, it was found that the hypotheses were so useful at the intermediate–level that it was appropriate to implement a coarse subset of

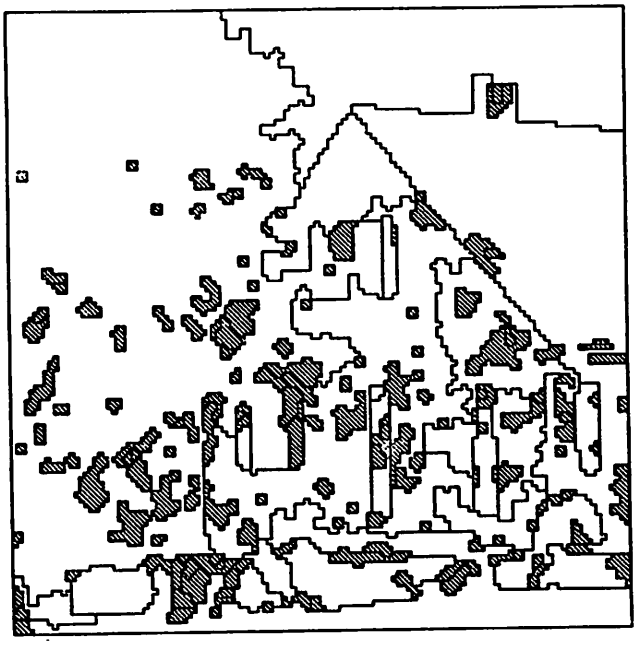


Figure 3.46: Merge–Potential Hypotheses with Gradient Constraint

semantic rules to be controlled from the intermediate level.

The rules used at this level were extracted from a rule–based hypothesis algorithm first developed by Williams [200], and later extended by Belknap et al. [15,80,166]. Through the experimental extraction of the feature characteristics of objects over a set of hand–labeled segmentations, the intermediate–level characteristics of objects (e.g. color, intensity, location, edge density) may be abstracted. The distributions of these feature measurements can then be used to define rules for the generation of hypotheses about the semantic labels which can be assigned to the region. The rules themselves approximate the conditional probability for a semantic label given the occurrence of a particular feature value. This value is then mapped into a range centered about the value 1.0 so that, as is the case with the other rules of GOLDIE, arbitrary combinations of rules may be formed.

The rules that are implemented at the intermediate level use the feature measurements of mean Itwor, mean Itwog, mean Intensity, centroid location, and

edge per unit area (of the Intensity image feature), to compute the value of the semantic hypothesis. These rules are stored on object nodes in ILTM, and may be applied as an intermediate–level process over a set of region tokens. It must be emphasized, however, that hypothesis values produced by these rules provide only crude approximations to the hypotheses that may be created by a high–level interpretation process, and are designed to be used only in the absence of such data. In general, the object information will be obtained from the high–level knowledge base of VISIONS when the two systems are closely coupled.

3.6 Lines from Zero–Crossings

The line extraction algorithms of the system are designed to create a set of intermediate–level line tokens based on evidence from image feature data. As will be described in Chapter 4, in the context of GOLDIE and VISIONS, a line may be minimally described by a pair of endpoints and a contrast value. Thus, the algorithms used for line extraction are quite dissimilar from the pixel based grouping processes that are used for region segmentation. GOLDIE employs two different algorithms for the creation of line segments in the intermediate–level database of the system. The first of these, zero–crossing based lines is a two–stage process using a low–level edge extractor and an intermediate–level grouping process. The second algorithm, which will be discussed in the next section, is a gradient–orientation based line extractor that computes the line data directly from the raw image data.

3.6.1 Edges from Zero–Crossings

The algorithm for the detection of zero–crossing based edges was developed by Boldt and Weiss [192] to provide accurate estimates of short line segments that could serve as the initial data for a collinear line grouping algorithm (described in Section 3.6.2). This motivation for the algorithm constrained the design of

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Figure 3.47: 3x3 Laplacian Mask

the algorithm in two ways: the extracted edges were required to be spatially restricted (edges should be present only in location where these is a potential for line grouping) and the orientation of the edges must be known with a high degree of accuracy.

The spatial constraint is implemented through the use of a Laplacian zero–crossing operator [125]. The image feature data is convolved with a 3x3 Laplacian (Figure 3.47), and the convolved data is thresholded to produce a set of contours that represent the zero-crossings of the operator. By using this size Laplacian, this stage of the processing demonstrates a reasonably good response to high-frequency data, while at the same time eliminating the placement of edge contours in areas of slow feature gradients. By restricting the extraction of short edges to the areas defined by this set of contours, the algorithm assures that edges will be produced only where significant discontinuities occur in the image feature data. As demonstrated by Haralick [90], however, the zero-crossing contours do not provide good estimates of the actual edge orientation at the points along the contours. Thus, for each pixel that intersects the zero–crossing contours, the gradient is computed from the image feature data. The assumption is made that the edge associated with each of these computed gradients has a length equal to one pixel, and that the magnitude and orientation are specified by the length and direction of the gradient vector.

Typical output of this process is demonstrated in Figures 3.48–3.50. For the image in Figure 3.48, the zero-crossing algorithm produces the contours shown

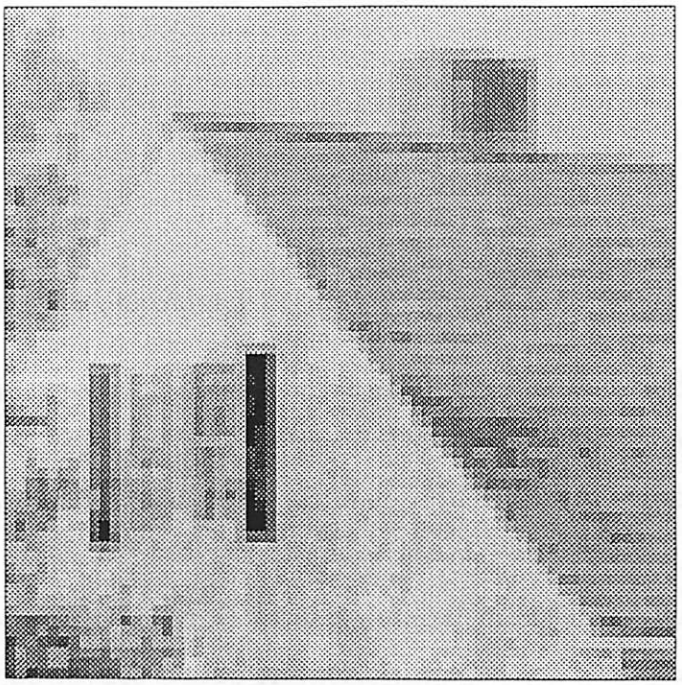


Figure 3.48: Image Data

in Figure 3.49. The actual placement of edges along these contours is shown in Figure 3.50, demonstrating the way in which this algorithm can produce a relatively small set of edge tokens[5] which, though short, are highly localized and quite sensitive to high–frequency data. Note, however, that an edge is created at each pixel that intersects a zero-crossing boundary so that an edge is created even if the gradient magnitude at that point is quite small. Figure 3.51 shows the subset of the edges from Figure 3.50 that have reasonably high contrast values.

3.6.2 Collinear Line Grouping

The Boldt algorithm [21,192] that has been implemented in GOLDIE for collinear line grouping uses an iterative hierarchical process to link adjacent edge or line tokens repeatedly until the longest possible line tokens have been created.

[5]The edge tokens are actually represented in the ISR as line tokens of unit length.

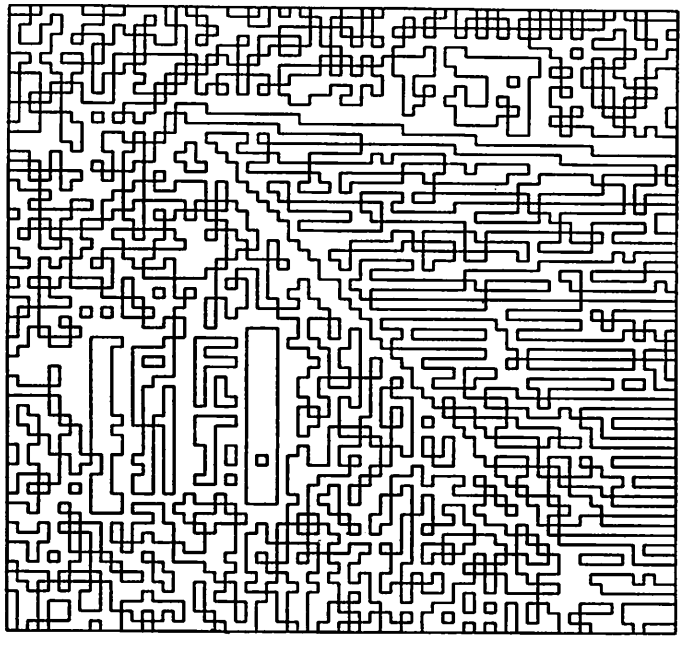


Figure 3.49: Zero–Crossing Contours

Each of the iterations consists of a link stage in which token pairs that meet the specified constraints are marked as potential merge candidates, and a merge step in which the best grouping for a given line token is identified; the line segment created by grouping these lines defines the structure of a new line token. The specific constraints for the linking step include: proximity of endpoints, similarity of line contrast, and collinearity. The actual threshold values used within each of these constraints are determined at the schema level of the system and specified as part of the task specification.

For each set of linked line tokens, a straight line is fit to the set of endpoints. A straightness measure

$$S = \frac{\sum_1^n d_i^2 \cdot l_i}{(n-2) \cdot s^2}$$

is computed for each set of linked tokens, where n is the number of endpoints, d_i is the distance from the i^{th} endpoint to the approximating line, l_i is the length of

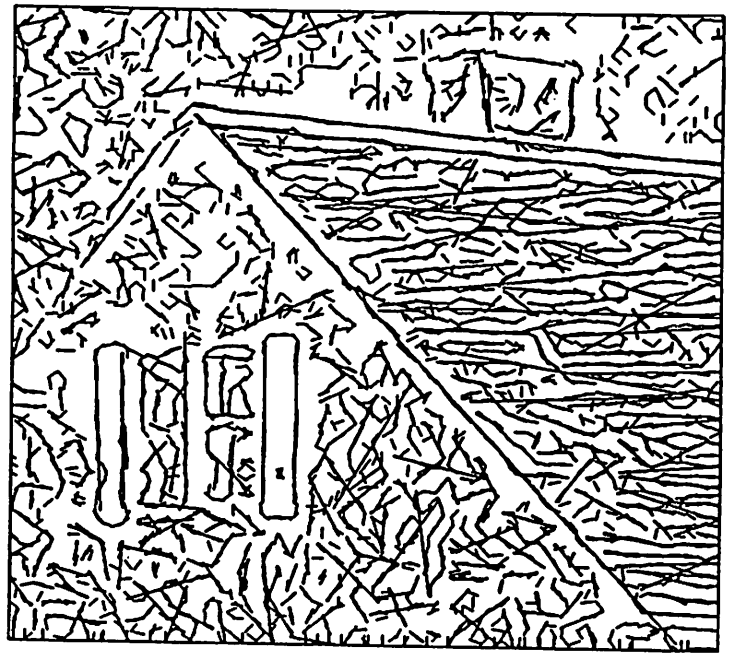


Figure 3.50: Edges Produced by Zero–Crossing Based Algorithm

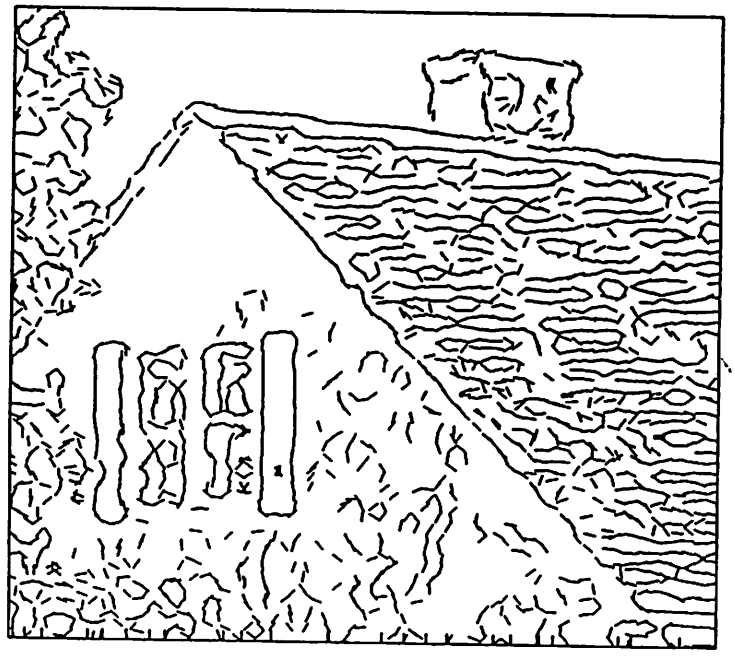


Figure 3.51: Edges from Figure 3.50 That Have High Contrast

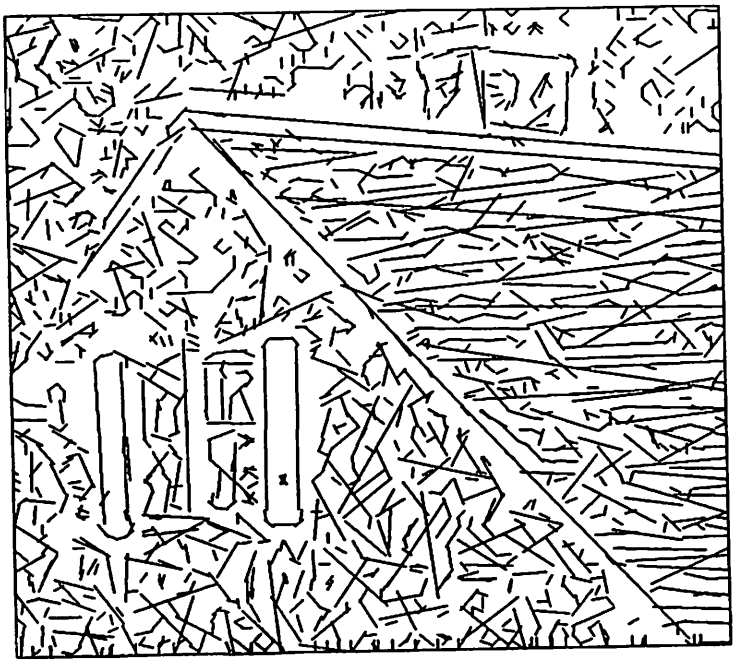


Figure 3.52: Collinear Line Grouping

the line segment corresponding to the i^{th} endpoint, and s is a prespecified scale factor. The computed line for which the straightness measure is a minimum, and below a constraint threshold, is selected as the merge of the component line tokens. A token is created for this computed line with a contrast determined as a weighted average of the contrast values of the component line tokens.

The hierarchical aspect of the grouping process has to do with the repeated application of the grouping process with increasing values of the scale factor s, so that longer and longer lines are created. Through the appropriate modification of the linking parameters, it becomes possible to constrain or relax this process to produce long lines with the desired specifications. Figure 3.52 shows the set of lines produced by the application of 12 iterations of the grouping process to the set of lines from Figure 3.50.

$$\begin{pmatrix} -1 & -1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & -1 \\ 1 & -1 \end{pmatrix}$$

Figure 3.53: $2x2$ Masks for Gradient–Orientation Based Lines

3.7 Gradient–Orientation Based Line Extraction

An alternative approach to the extraction of straight lines, based on estimates of the local image gradient, was developed by Burns [33,34]. The underlying assumption for this algorithm is that lines in a scene will be represented by a connected set of image pixels that demonstrate similar gradient orientation on the three–dimensional surface defined by the individual pixel values of a given image feature.

In the implementation of this algorithm, the orientation and magnitude of the intensity gradient at each point in the image are computed from the horizontal and vertical differences in feature values between adjacent pixels according to the $2x2$ masks [88] shown in Figure 3.53. The orientation is then partitioned into buckets of 45° width, and a connected components algorithm is run to label the *line support regions* (regions of constant gradient orientation). Each of these support regions represents a line in the image. Because this quantization has the potential of fragmenting support regions in which the gradient orientation overlaps a partition boundary, an additional quantization and connected components process is run on the data with the partition buckets rotated 22.5° from the original to provide a second set of line support regions.

A least–squares algorithm is then used to fit a plane that approximates the intensity surface of the image feature for each support region. The gradient direction of this plane defines the orientation of the line to be formed from the support region, and the steepness of the plane describes the contrast. The actual

placement of the line within the support region is determined by intersecting this plane with a horizontal plane representing the mean value of the image feature over the support region. The line produced by this intersection is clipped against the boundary of the support region to create a line segment that is represented as a token in the ISR.

Since each pixel in the image has contributed to two lines according to the two quantizations of the gradient orientation, a great deal of this line information is redundant. Therefore, a form of resolution is performed in which each pixel "votes" for one of its two support regions according to the length of the line created from that region. Lines resulting from support regions that receive votes from a majority of the component pixels are retained, while the other lines are deleted.

Figure 3.54 shows the lines produced by this algorithm on the image from Figure 3.48, while Figure 3.55 shows the subset of these lines which exceed a contrast threshold. Since this algorithm is not restricted to producing lines on zero–crossings, a much larger number of lines are produced by this algorithm than by the previous algorithm. While it is true that most of these lines do not correspond to significant object boundaries in the image, they do convey a great deal of information about the textural characteristics of the feature data. Figure 3.56 shows the gradient–orientation based lines produced from a larger image (Figure 1.2), and demonstrates the way in which the characteristics of the lines vary across the different objects present in the image.

This observation led to the definition of the short line image feature (Slfeat) described in Section 3.1.2. This image feature is designed to differentiate between the areas of the image that contain many short high contrast lines and those which contain low contrast or long lines. Given the gradient orientation based line for which a given pixel is a member of the line support region, the initial estimate of the image feature value defined at that pixel is the ratio of line contrast to line length. A $3x3$ Gaussian is convolved with this set of initial estimates to produce

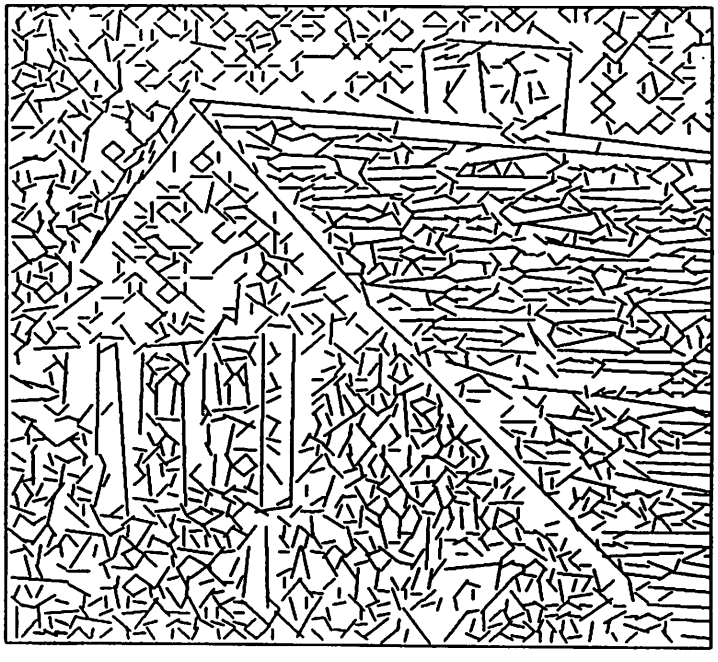


Figure 3.54: Lines Produced by Gradient Orientation Algorithm

the image feature shown in Figure 3.57.

Note that since these lines are represented as tokens in the ISR, it is also possible to use the collinear line grouping algorithm from Section 3.6.2 on these lines as well.

3.8 Line Segmentation

Given the presence of both region data and line data in two different representations, there is an obvious need for a mechanism to integrate the information represented by these two types of tokens. Line segmentation processes are those which are designed to produce this integration through the modification of region tokens based on the possible existence of a (set of) lines. Through these processes some of the ambiguity and redundancy introduced by these two types of intermediate level representation may be resolved by a low–level process. Two different

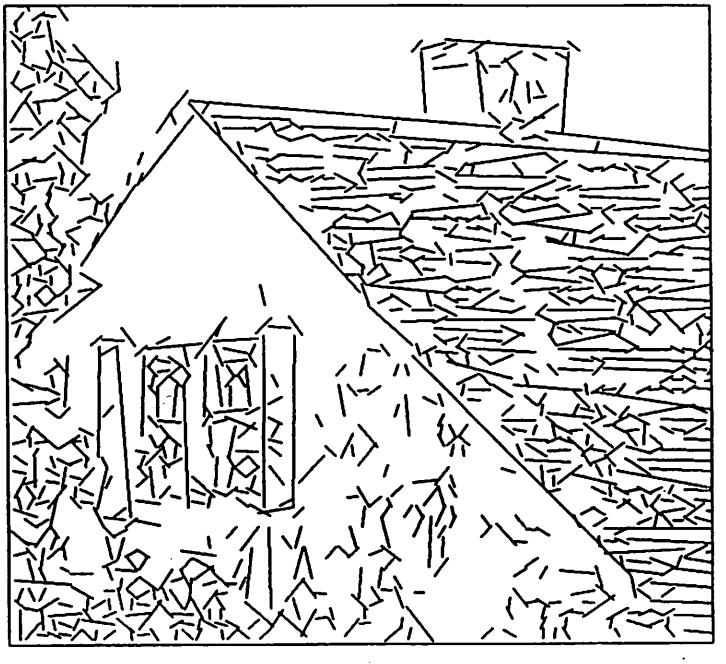


Figure 3.55: High Contrast Lines Produced by Gradient Orientation Algorithm

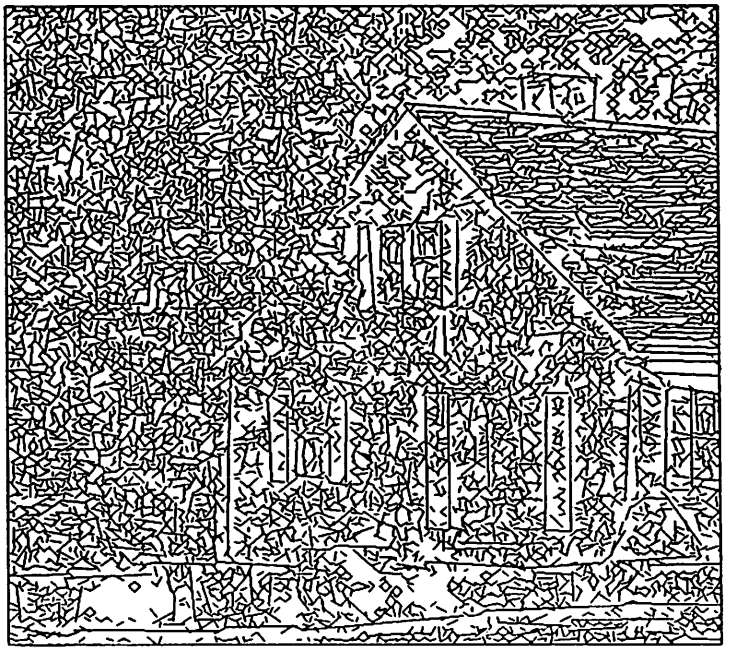


Figure 3.56: Lines Produced by Gradient Orientation Algorithm

Figure 3.57: Slfeat Image Feature

algorithms for line segmentation have been implemented in GOLDIE in order to serve two different types of goals. The first of these involves the use of lines to directly split intersecting regions, while the second involves the specification of a particular line that will be used by an instance of the region–segmentation schema in an attempt to invoke a segmentation process that will produce regions with boundaries on, or near, the line. boundaries on the line.

3.8.1 Direct Line Insertion

Direct line insertion is designed to be used in the case that there is strong belief in the existence of the lines, and when the specifications of the lines are known with a high degree of accuracy. This process would be used, for example, in the case when the set of lines had been produced through a line extraction process, and were therefore known to be present in the raw image data. The

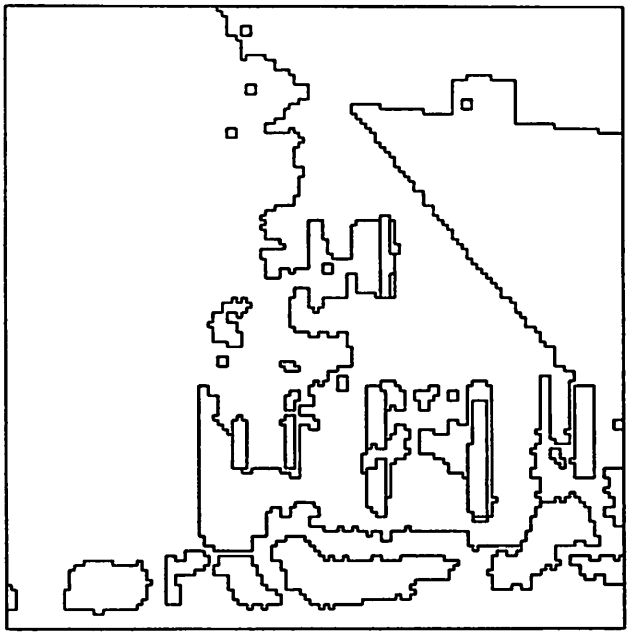


Figure 3.58: Initial Region Segmentation for Direct Line Insertion

purpose of the direct line insertion process is to provide a set of regions that have been derived from two quite different, but complementary types of data. Regions have been constructed by what is essentially a grouping process, while lines have been constructed by some form of differencing operation. Since there are many type of image discontinuities that are detected by one or the other, but not both, of these processes, it is often useful to combine the two sets of data to produce regions whose boundaries reflect both types of processing.

The algorithm for this process is quite straightforward. Given a set of lines and a set of region tokens, as in Figures 3.58 and 3.59, the set of region tokens intersected by the lines is determined (Figure 3.60). A new image plane containing the boundaries (represented as horizontal and vertical edge elements) of these region tokens is then created. The lines are then drawn in as additional boundaries of this segmentation in a two stage process. In the first phase, each of the lines is drawn into the horizontal and vertical edge representation using

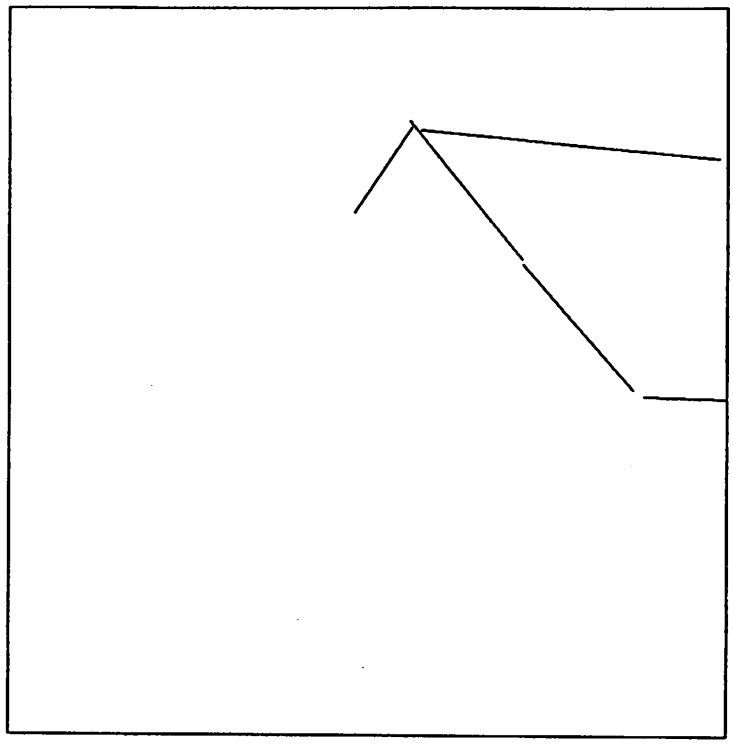


Figure 3.59: The Set of Lines for Direct Line Insertion

a standard Bresenham [25] vector algorithm. However, simply drawing the lines into the boundaries is not sufficient. Since these lines are intended to split the regions they intersect, a second pass is made to extend these lines to force them to terminate on existing boundaries. Each of the lines is extended from each endpoint until a length threshold (expressed as a percentage of the original length of the line) is exceeded, or until an existing boundary is reached. Because this extending line may run parallel to an existing boundary in the segmentation (perhaps due to a one or two pixel displacement between a region boundary that was created by a region segmentation process and a line created by a line extraction process), intersection is forced if the line being extended falls within two pixels of an existing boundary.

This extension procedure guarantees that when a connected components process is run to create regions from the augmented set of boundaries, a new segmentation will be formed that incorporates the line data in the region segmentation.

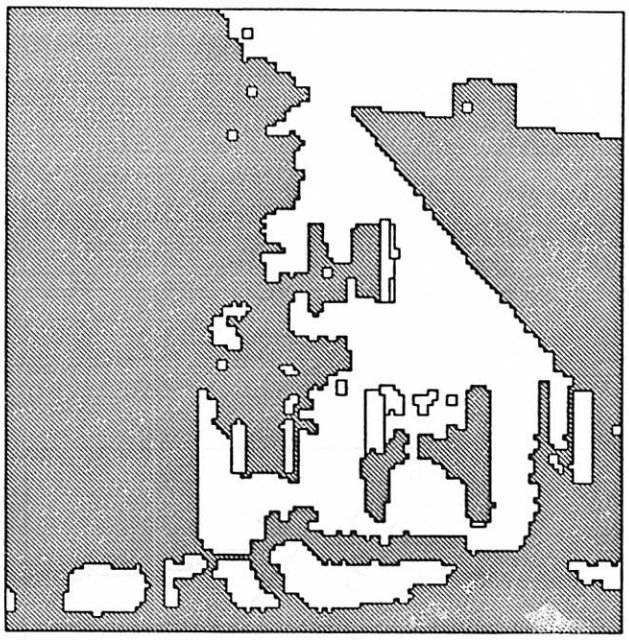


Figure 3.60: Intersecting Regions for Direct Line Insertion

Without this mechanism, lines that did not intersect any existing boundaries would be lost in the connected components process. Minor errors of overfragmentation are generally less expensive than the regeneration of missing boundaries. However, since the extension threshold is a parameter of the line segmentation process, variations may be made in this value depending on the nature of the lines to be inserted.

Figure 3.61 shows the result of the direct line insertion for the sets of regions and lines referenced above. Note that although fragmentation has occurred in the area of the roof line, where the line is closely parallel to the region boundary, this error can be corrected later through appropriate region merging. The intent of this process is to guarantee that most of the inserted lines will be represented as region boundaries in the resulting segmentation.

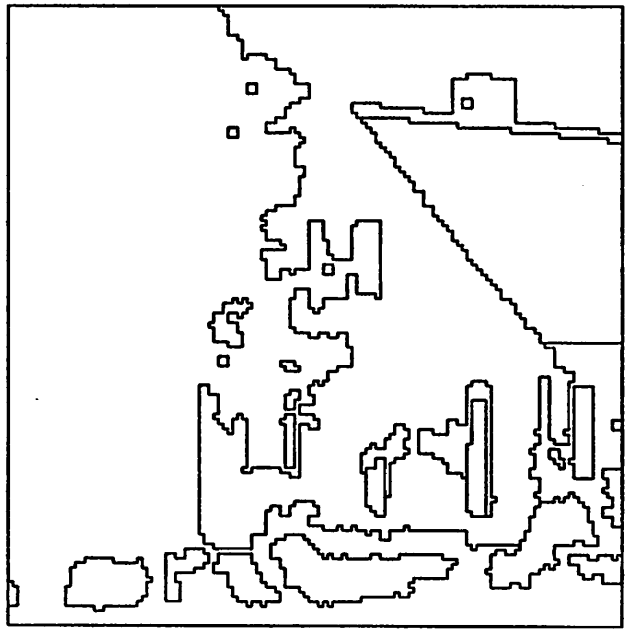


Figure 3.61: Region Segmentation Resulting from Direct Line Insertion Process

3.8.2 Line–Based Segmentation

A different scenario for the use of lines to modify a region segmentation would occur if a high–level interpretation process concerned with shape were to make a hypothesis that a line should possibly exist in an area of the image. In this case, direct insertion would not be desirable (i.e. the hypothesis might be incorrect). Therefore, a second line–based segmentation algorithm has been implemented in the GOLDIE system to deal with this type of situation. The essence of this algorithm is that the characteristics of image features on either side of the hypothesized line can be examined by a low–level process, and then this information can be used by a schema process to select the image feature for region segmentation. The image feature that exhibits the largest difference across the line would be expected to find a region boundary near the line if the underlying data supports such a boundary. The major difference between this process and the direct line insertion is that the previous algorithm assumed accuracy of line information, while this process treats the line data as inexact. The intent here is to provide a form of control over the region segmentation process so that the regions created will potentially have boundaries that will approximate the specified line. In the following example, we hypothesize a situation where the line between house wall and sky which was used in the previous example could not be found in the raw image data (Figure 3.62). In such a situation, it would be reasonable to extend the long, high contrast line at the top of the roof and use the extension of this line as an approximate indicator of the actual boundary between sky and house wall (Figure 3.63).

The low–level components of this line–based segmentation process involve feature measurement and region segmentation. In order to measure the feature distributions, a temporary segmentation is created which contains the region token to be segmented. The line is then inserted into this temporary region to produce two (or more) new region tokens (Figure 3.64). Since this process produces a set of regions which are divided by the proposed line, the feature measurement

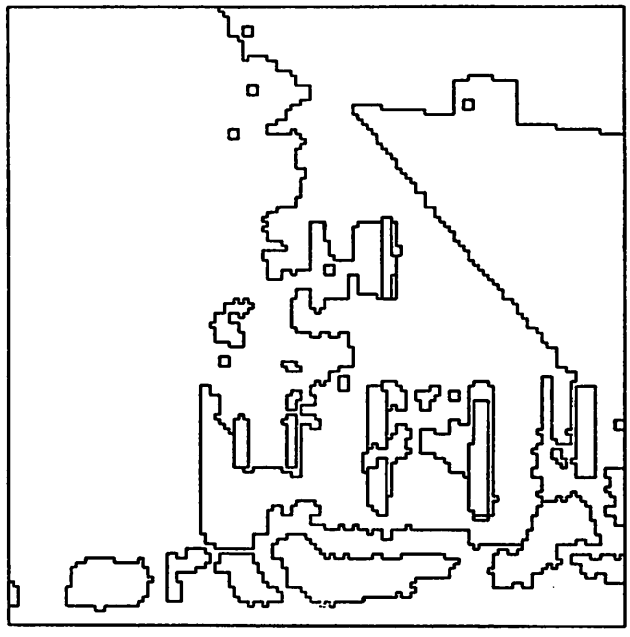


Figure 3.62: Region Segmentation for Line–Based Segmentation

tools of the system may be used to measure the characteristics of image feature distributions about the line. Once these measurements are complete, the regions created in this process may be deleted from the database of the system so that no artifact of the measurement process remains in the set of region segmentations.

Based on the data produced, a schema instantiation can then select appropriate image features and region segmentation algorithms from the sets mentioned in Sections 3.3 and 3.1 to specify the region segmentation process that is then used to resegment this region. As shown in Figure 3.65, the measurements obtained from the temporary region tokens allow the line–segmentation process to choose an appropriate image feature (Red) and an appropriate segmentation algorithm (global one–dimensional histogram clustering) that accurately locate the scene boundary that was approximated by the hypothesized line.

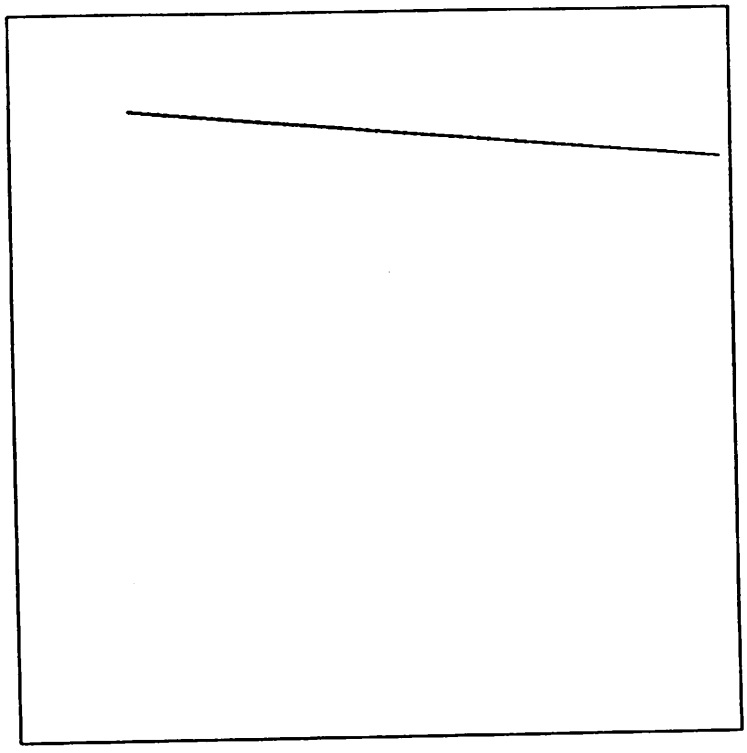


Figure 3.63: Line for Line–Based Segmentation

3.9 Low–Level Process Controller

The Low–Level process controller is implemented as a set of parameterized LISP functions running within the context of the VISIONS system. These functions form the interface between the high and intermediate–level control processes and the image operators and ISR data access routines. The intent of this interface is to isolate the specification of low–level processes from the actual implementation of those processes. The controller enforces a consistent interface protocol for the activation of low–level processing tasks, permitting the schema representation of low–level processing to be expressed in a manner independent of the implementation or even the processing environment (e.g. image processing software or computer) of those tasks. Thus the addition, deletion, or modification of low–level processing tasks may be accomplished with minimal modification of the schemas that control the tasks.

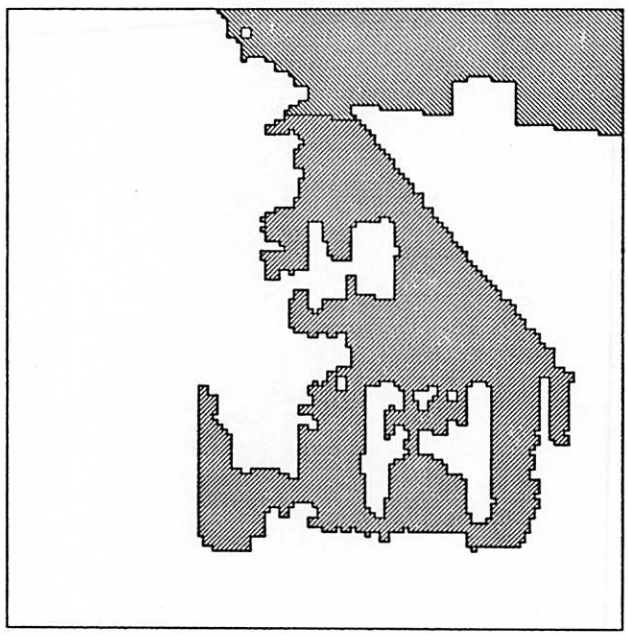


Figure 3.64: Temporary Regions for Line–Based Segmentation

With respect to the intermediate–level data structures that will be discussed in the next chapter, the controller is also responsible for the maintenance of the structures related to the region segmentation nodes and the line–extraction nodes that encode the relations between tokens and the processes by which they were created. This data structure provides the information necessary for backtracking if the schemas decide at any point that specific segmentation processing should be "undone". Thus, if several different attempts were made to segment an area of the image, but only one of these was found to be acceptable, this mechanism allows the deletion of the tokens produced by the unacceptable segmentation processes.

The types of routines which make up this interface fall into several classes:

- **Get–token–value** - Access, or if necessary compute, a feature value of a token according to the function specified in the ISR.

- **Post–token–hypothesis** - Post the value of hypothesis about a token in

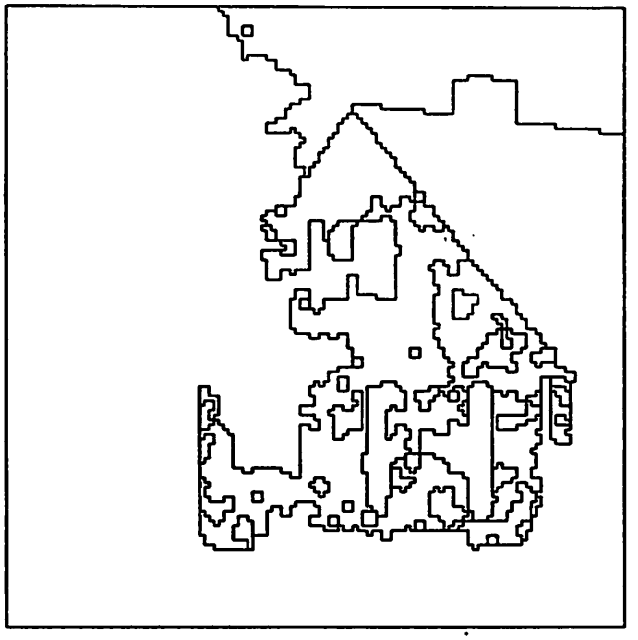


Figure 3.65: New Regions Produced by Line–Based Segmentation

the ISR.

- **Apply region segmentation algorithm** - Invoke the named algorithm with the specified parameters to produce intermediate–level region tokens. For example, in the case of region segmentation the parameters are region token, image feature, segmentation algorithm, segmentation sensitivity, and postprocessing algorithm.

- **Apply line–extraction algorithm** - Invoke the named algorithm with the specified parameters to produce a line–extraction node.

- **Apply grouping process** - Group tokens to create a new token and make necessary modifications to the graph relations.

- **Apply knowledge source** - Apply a named process to image feature data which is defined by the bitmap of a token, and return a value (e.g. compute

the modality of the histogram of a specified image feature across a region).

3.10 Conclusion

The various low and intermediate–level processes that have been detailed in this chapter demonstrate the wide variety of processing tasks that can be invoked within the goal–directed control paradigm. Although this set of processes will be shown to provide a great deal of effectiveness with respect to the interpretation process in the domain of natural scene images, we believe that many other low and intermediate–level algorithms could also be useful to a complete image interpretation system.

The intent of this chapter has been twofold. First, we have described the algorithms that have been used to generate the data in this dissertation. Second, we have attempted to demonstrate how the architecture of GOLDIE provides the flexibility and extensibility for low and intermediate–level processing in a goal–directed system. The system itself is not in any way restricted to these particular tasks. Through the use of the low–level process controller, the image and token processing is separated from control processing. If it becomes necessary to incorporate additional types of low or intermediate–level processes, the extensions to GOLDIE are straightforward. By constructing these processes so that the output data is represented as tokens in the intermediate–level database, these new processes can become available to the schema–based control mechanisms of the system.

Chapter 4

Intermediate Level Data Structures of GOLDIE

4.1 Introduction

As was demonstrated in Chapter 1, the data structures of the intermediate level of vision processing are a critical element of the image interpretation process in that it provides the interface between the data–oriented low–level processes, and the semantically based high–level processes. Because these two types of processes operate in quite different domains, the data structures which are typically used in these processes for the representation of data and control information do not allow for efficient communication.

Since interpretation processes utilize the data from segmentation processes, some form of data compatibility is required between the processes at the two levels. Typically the compatibility is unidirectional in the sense that the high–level processes are designed to understand the data output of the low–level processes. Few mechanisms exist by which higher level processes may provide information which can guide low–level processes. As an example, the $2\frac{1}{2}D$ sketch representation of Marr [127] is an intermediate level data structure which exhibits this unidirectional communication pathway. The $2\frac{1}{2}D$ sketch is a viewer-centered representation that uses surface primitives of one (small) size. It includes a representation of contours of surface discontinuity, and has enough internal computational structure to construct its descriptions of depth, surface orientation, and surface discontinuity. The surface primitives, however, have no semantic identity. They are constructed by low–level operators, and any high–level interpretation must perform grouping operations that integrate over sets of the surface primitives.

This representation contains no data structure that can be directly manipulated by both the high–level and low–level processes.

The specification of the intermediate–level structure is complicated further by the need for a coordinated representation of control state at the intermediate level. Through a consideration of both of these issues, a number of data structure requirements become apparent.

4.2 Data structure Requirements

The requirements for a complete intermediate level data representation are extensive. Briefly summarized, they are as follows:

- **Coherent representation.** The system should be coherent in the representation of entities: regardless of whether the entity is high or low–level, it should have the same external appearance in the system. The coherence allows efficient bidirectional communication, since the message passing and control mechanisms that access the data become independent of the data structures themselves. In a sense, this protocol is analogous to that of the object oriented systems such as Smalltalk [72,107] or Steamer [63,64]. In this context, object-orientation means that an entity of the system, be it an image abstraction, knowledge structure, or interpretation hypothesis, is defined strictly in terms of external appearance. It becomes the obligation of any control process that attempts to utilize this entity to understand its internal structure.

- **Dynamic data relations.** The data structures should be highly relational. As has been shown, the segmentation/interpretation process is a highly dynamic one. The data structures must support this dynamic nature. We need to avoid any representation that requires a complete and fixed hierarchical representation of the entities in the system. Relations between the entities of the system (e.g. spatial or hierarchical) must be

flexible so that we may have the capability for experimentation with new relations, and so that we may promote dynamic control of the data abstraction process.

- **Dynamic control.** In the same manner, the data structures must support the concept of dynamic control of the interpretation process. Both the data abstractions (e.g. tokens) and the knowledge abstractions (e.g. hypotheses) must be realizable in this data structure. This includes a representation of the current control state of the system as well as a representation of the processing history for all of the entities in the system.

- **Support for parallel processing.** Parallel (or simulated parallel) processing of the image must be supported in the representation.

- **Flexibility and extensibility.** A final requirement, critical to the research purposes of this system, is that the data representation be dynamic and extensible. We require the capability to associate any fact of information with an entity at any stage of the processing. For fixed domains and sets of algorithms, this requirement may be relaxed for efficiency, but the flexibility and extensibility are critical to the early development of a system.

4.3 Semantic Network Representation

In view of the requirements discussed above, we have placed a strong emphasis on the design of the data, knowledge, and control representations for the GOLDIE system.

The underlying structure used in this representation is that of a semantic network: a graph consisting of nodes and directed arcs [12]. Such a structure has been used extensively in high level data representations [23,202], but has been used infrequently in low or intermediate–level representations.

In GOLDIE, the entities of the intermediate–level representation are each represented as a graph node. The various classes of nodes that can be represented in the semantic network of GOLDIE include: images, image feature data, low–level process descriptions, intermediate–level process descriptions, intermediate–level image abstractions (tokens), hypotheses, and goals. A node has a unique name which identifies it to the system; by convention the name of the node is the concatenation of the name of the node class with a unique identifying number. A node additionally has a value, which in this implementation is expressed as an attribute list: a list of name/value pairs. Thus, any fact about an entity may be expressed as an attribute of the node.

The relation between any two of these nodes is represented by an arc. The arc, or relation, terminates on the two nodes and has a specific direction (startpoint and endpoint). The names of arcs are not necessarily unique; our convention here is for the arc name to expresses the nature of the relation between the two nodes. Thus a complete specification of an arc requires two nodes, a direction, and a name. The arcs can also have a value, and in GOLDIE a set of values may be associated with a particular arc, again through the use of an attribute list.

4.3.1 Utility of the Network Representation

This network representation provides a straightforward and concise form for data storage for the GOLDIE system. There is no inherent characteristic of the representation that violates any of the specified requirements, and the approach greatly clarifies the interface between the high and low level processes involved in the image interpretation process.

Since a node is a named entity that contains a list of name/value pairs, it is possible to encode an arbitrary set of properties for a given data abstraction. Explicit arcs are used to express any relations to other nodes. These arcs also provide the ability to express information that pertains to an entire class of nodes on an ancestor of the class, rather than expressing the same information on each

node of the class. This mechanism is similar to the general semantic network concept of taxonomic inheritance [60], but is used in a somewhat more restrictive sense in the GOLDIE system. Our network does not have a strict hierarchical taxonomy, since many relations that we need to express are not hierarchical in nature. Thus, class inheritance is not a general attribute of the system, but as will be seen, inheritance can be made explicit in those cases where it can be useful.

Coherent data representation throughout the interpretation system is assured by the fact that all data structures of the intermediate and high level processes are uniformly represented as nodes in the network. This gives a unified structure which requires a single set of functions to provide data access to all levels of the system. This same network structure also provides a representation in which the explicit control structures of the system are represented. Each intermediate–level control process (schema instantiation) or goal of the system is represented by a node that expresses the set of variable bindings that define the particular instantiation. Each schema instance is linked to a specific goal node, providing a dynamic control structure in which the number and type of intermediate–level control processes are determined by the goal state of the system. The actual details of this interaction will be introduced in Section 4.8 and then examined in detail in Chapter 5.

The intermediate level data representation also allows another form of dynamic control with respect to the implementation of the concept of *active values* [60]. This form of control deals with the dynamic application of low-level processes such as feature calculation. Since we construct node and arc values as arbitrary LISP attribute-list structures, selected attributes of nodes may actually be LISP functions for the calculation of an attribute value. By binding a function into the name/value pair of an attribute list, a value is computed only if that data value is accessed. This active value representation allows the system to operate as if all possible information has been precalculated prior to processing. This methodology permits a great deal of flexibility in the design of the knowledge

sources that utilize computed image data.

Spatial relations are conveniently expressed through the combination of positional data values on nodes, and the use of arcs to represent adjacency or proximity. Similarly, hierarchical data relations are expressed through the taxonomy of the semantic network.

Processing history is expressed through a combination of node attributes and arc relations. Specific facts, such as the time at which a node was created, are expressed as an attribute of the node. More complex relations, such as the specification of the segmentation algorithm which produced a particular region, are expressed through the specification of segmentation nodes that are linked to the region and which have a set of attributes describing the segmentation algorithm.

Finally, we note that this network representation provides an extremely flexible and dynamic representation for image structures. Any entity which has been constructed from the image may be represented in the system through the construction of a new node in the network. As new structure types are designed it is sufficient to define the node type that specifies the structure. All relations to existing structures in the system may then be expressed as arcs to the appropriate related structure nodes.

4.3.2 Implementation of the Semantic Network in GOLDIE

This network structure has been implemented in the graph processing language GRASPER [124], which is a LISP based graph processing language developed at the University of Massachusetts at Amherst. This language supports the representation of nodes, directed arcs, and spaces which partition the graph. Since the nodes and arcs are themselves LISP objects, the graph may be manipulated either through the GRASPER primitives (e.g. set-of-inpointing-nodes, set-of-inpointing-node-given-an-edge-name, set-of-nodes-in-a-space, etc.), or with standard LISP operators. This flexibility allows us to access the data structures from any level of our interpretation system. An additional benefit of the

GRASPER implementation is that it provides an efficient mechanism by which states may be saved and restored, providing experimental flexibility. The disadvantage of GRASPER is that it is slow, but this was not considered to be of significant importance in the first implementation of GOLDIE.

4.4 Intermediate Symbolic Representation

Although the semantic network formalism provides all of the flexibility and generality necessary for intermediate–level data representation, the one disadvantage is computational complexity. Especially in the case of lines, where there may be five to ten thousand lines extracted over a ($256x256$) image, the cost of creating nodes for each of the tokens may be prohibitive. Even with respect to regions, storage of a large number of features on the attribute list of the region node entails significant LISP overhead.

It was therefore necessary to augment the semantic network representation for the storage of token data with an existing database mechanism in the VISIONS system known as the ISR (Intermediate Symbolic Representation) [83,181]. This mechanism is designed for efficient and economical access to information about image tokens. Although the introduction of this mechanism violates the concept of a coherent network representation of the intermediate–level data, the two mechanisms have been integrated in such a fashion as to completely satisfy all of the data structure requirements while significantly improving system performance[1].

The ISR mechanism has been written in C rather than LISP in order to provide an efficient mechanism for the storage of large amounts of intermediate–level data. Since the nature of intermediate–level data typically entails a relatively small number of token classes, each with a large number of members, management

[1]Extensions are currently being made to the ISR that will provide efficient storage of token–token relations in the same C structures. When this is accomplished, it may be possible to store all token information in the ISR and thereby remove this dual representation.

of this data can be performed effectively without the complete generality of LISP.

The representation of a token within the ISR is as a named object that has a bitmap representing its two–dimensional spatial extent, and a list of attributes that describe the image characteristics of the token. This list of features is expressed as a *lexicon.* The specific lexicon associated with a token defines its class; in GOLDIE, the two lexicons **Region** and **Line** define the two classes of tokens known to the system. The lexicon defines the name and datatype for each feature that can be measured, and optionally a function that can be used to compute that feature if no value has been stored. A description of the actual definitions of the region and line lexicons will be presented in Sections 4.6.4 and 4.6.5.

The active value mechanism for token feature values expressed through the compute function in the lexicon feature definition allows the same generality for ISR tokens that is provided in the semantic network representation. These functions are defined in LISP, so that the full functionality of GOLDIE may be utilized in the calculation of these values. If a feature value of a token is expected to be required (e.g. the size of a region token) it can be calculated within the ISR and specified at the time of token creation. If, on the other hand, it is unknown if the feature value will ever be accessed, it can be left undefined so that it will be calculated only on demand.

Another useful characteristic of the ISR lies in the methods of accessing tokens. Tokens in the ISR may be accessed either by name (e.g. Region-00001) or by value (e.g. which of a given set of region tokens are larger than 20 pixels?). This capability allows rapid associative access to relational information about tokens in the database. For example, the ISR makes it trivial and efficient to ask for all line tokens in the database which intersect a given region token and which have length and contrast values which exceed given thresholds. As will be shown in Chapters 5 and 6, this type of request is made quite often by the intermediate–level schemas of GOLDIE.

The ISR is a very general method for the storage of intermediate–level data

(i.e. data which is in registration with the image), providing the capability for dynamic creation of arbitrary tokens and lexicons that could permit a hierarchical organization expressing the relations between tokens. Such a structure would conceivably be capable of expressing any token relations which are currently stored in the semantic network of ISTM. Within GOLDIE, however, the representation is used only to represent primitive region and line tokens. The reasons for this choice are partly historical since the ISR did not exist when GOLDIE was initially designed. More importantly, however, the use of a semantic network to express the organization of tokens in the ISTM allows a more general flexibility in the expression and use of new and experimental types of relations.

4.5 Intermediate–Level Representation of GOLDIE

The data structures of GOLDIE (Figure 4.1) may be described in terms of four primary modules:

- **ISTM** - containing intermediate–level tokens, images, low-level processes, and hypotheses,
- **ILTM** - the representation of explicit intermediate–level knowledge,
- **Goal Blackboard** - the representation of goals to be processed by GOLDIE, and
- **Schema Instantiations** - the representation of the control state of the system.

The following section presents a description of the dynamic structures of ISTM, including region and line tokens, hypotheses, low–level process descriptions, and image structures. The static knowledge structures of ILTM will be discussed next. Finally, there is an introduction to the control representation of the schema instantiations and goal blackboard.

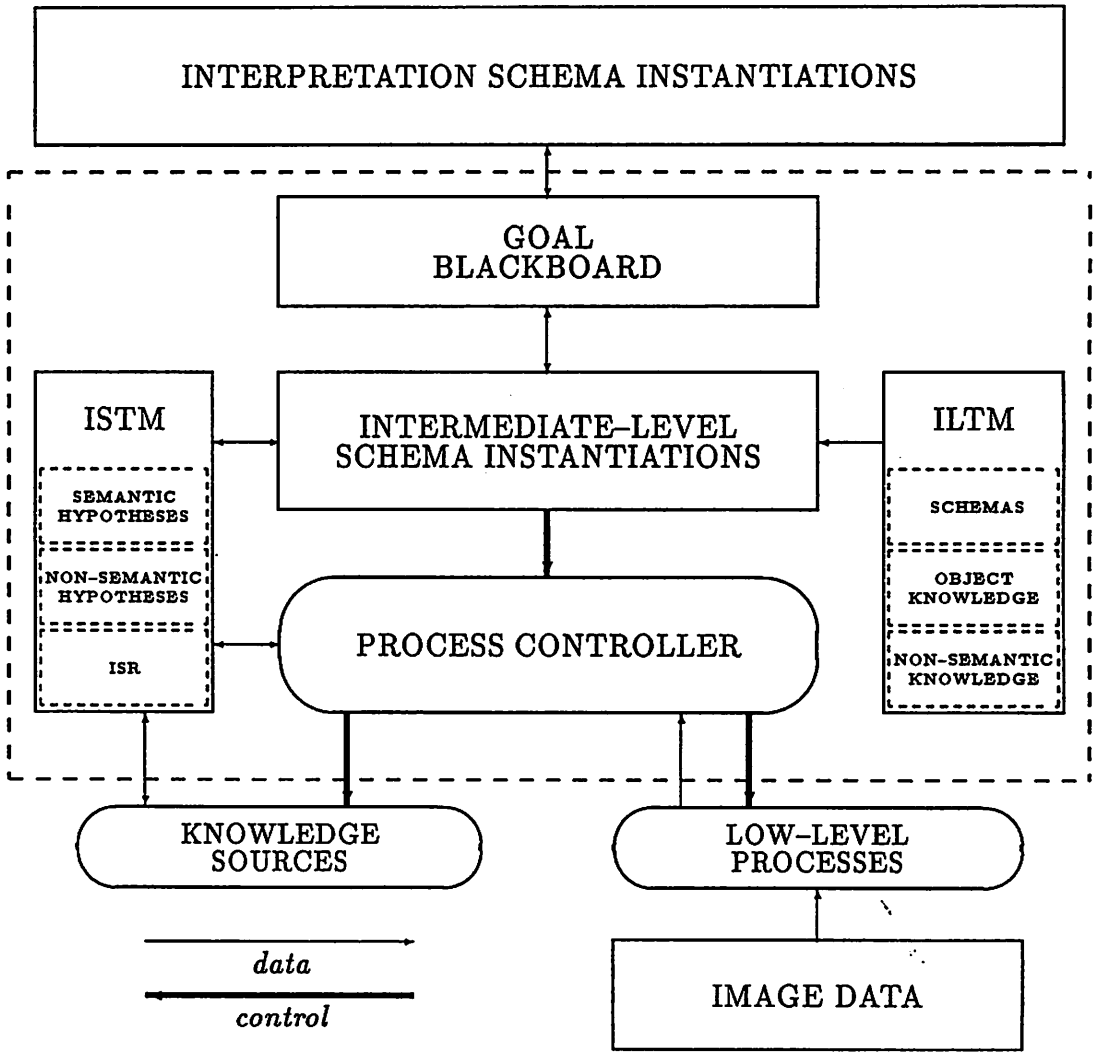


Figure 4.1: Overview of the GOLDIE System

4.6 Dynamic data structures of ISTM

The data structures of ISTM provide explicit representation for five primary entities: images, region tokens, line tokens, process descriptions, and hypotheses. Since these structures are crucial to both the conceptual organization and implementation of the GOLDIE system, they will each be described in detail.

4.6.1 Image Data Structures

Image data is represented in the VISIONS system as an array of numeric image values. This array is stored in a *plane* data structure as a row-major data array that is accessible by both the Fortran or C based image processing modules and the LISP control modules of the system. This storage structure encodes image size, resolution, data-type, and descriptive text in a structure associated with the data array, and permits a consistent and flexible access protocol for all image data of the system. Since image operators are conceptually specified as modules that are applied in parallel to each pixel of an image, this type of data storage allows the system to operate in a manner independent of image size and data-type. Access to the data is controlled through a set of access functions that allow an operator to address a given pixel for the specified image. The pixel address may be specified either in a relative or an absolute manner. Relative addressing may be specified with respect to a given pixel in the same image or hierarchically with respect to the location of a pixel in a separate image. Absolute addressing specifies a pixel relative to the image origin.

In forming an intermediate level data representation for image data, there are several factors to consider. The raw image data may be either intensity (gray–scale), RGB, or some other form of sensory data. Additionally, the image may be viewed at any one of the multiple levels of the processing cone [81]. These factors introduce the need for a structure representing the concept of an image: an intermediate–level data structure that is a separate entity from the

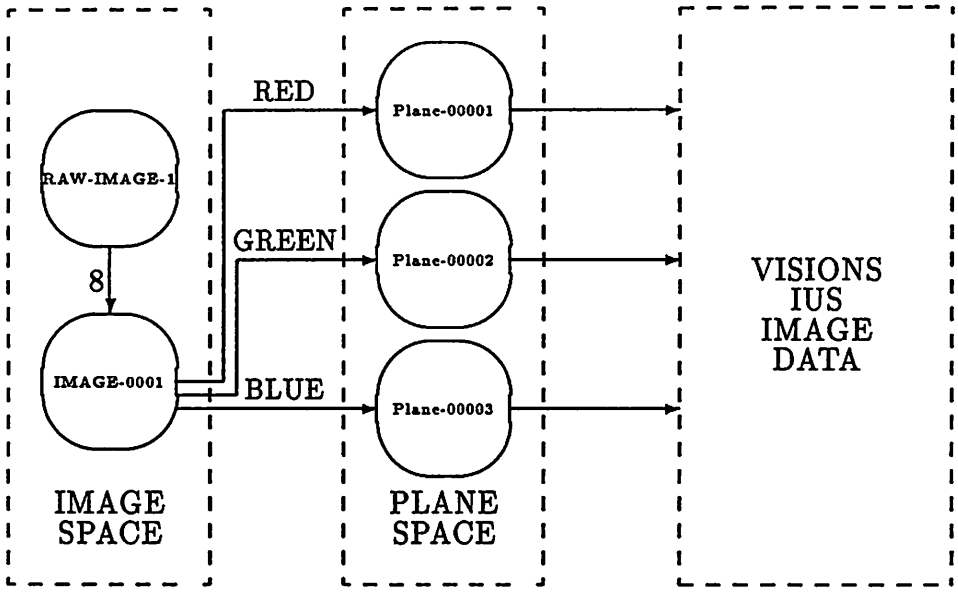


Figure 4.2: Image Node Structure

data (planes) that comprise the image (Figure 4.2). This distinction is preserved in the GOLDIE semantic network through two partitions known as *image* and *plane* spaces. As an (intensity or RGB) image is brought into the system a `Raw-image` node is constructed in this space which encodes the level of resolution and component data arrays of the image being created. Following this, a separate `Image` node is created. This node is linked to the `Raw-image` node with an *Rawimage-Image* arc that has a value indicating the level (with respect to the processing cone) of the data planes; in Figure 4.2 this is indicated by the arc with the label "8".

The data planes themselves are each represented by a separate `Plane` node in the *Plane* space. Each of these nodes stores the index, level, image feature name, and other bookkeeping information regarding the VISIONS system plane it represents. This (set of) `Plane` node(s) is linked to the `Image` node by an *Image-Plane* arc with a value indicating the image feature represented by the plane.

If a new level of this information is desired, a new Image node is created and linked to the Raw-image node through an arc specifying the new image level. The data planes corresponding to the raw-image are then reduced or projected through the processing cone and linked to this new Image node.

Conceptually, this means that each level of the processing cone is occupied by a (set) of data planes created through reduction/projection of the original data. However, the actual data is created only if requested. For example, consider the case diagrammed in Figure 4.3. Here the raw-image has been specified as a RGB image at level 8 (256x256) of the processing cone. If some Knowledge Source now requests the level 4 reduction of the red plane in Image-00001, a new image node is automatically created for level 4 (Image-00002), and a reduction operator is applied to the red plane of Image-00001 (Plane-00001), creating Plane-00004[2]. This Plane node is linked to Image-00002 and returned to the requesting Knowledge Source.

Image feature planes are data arrays that represent some form of transformation which preserves the spatial relations of the original data. Such a transformation may be a simple linear transformation such as a scaling, or a more complex transformation such as a segmentation region labeling. Intrinsic images in the sense of Tenenbaum and Barrow [13,14,183] would also fall into this category. In order to provide a representation for this data, a network partition called *Feature* space has been created. There is a node in this space for each of the feature transforms known to the system. The name of the node is the name of the feature, and the value of the node contains the function that is used to calculate the feature from the RGB or Intensity data.

This structure enables the following protocol for computed image data access. When a knowledge source requests a particular feature for a specified Image node,

[2]Whenever a new Image node is created, the system automatically projects or reduces the original data and creates the appropriate Plane nodes. Thus, the Green and Blue planes have also been created at level 4.

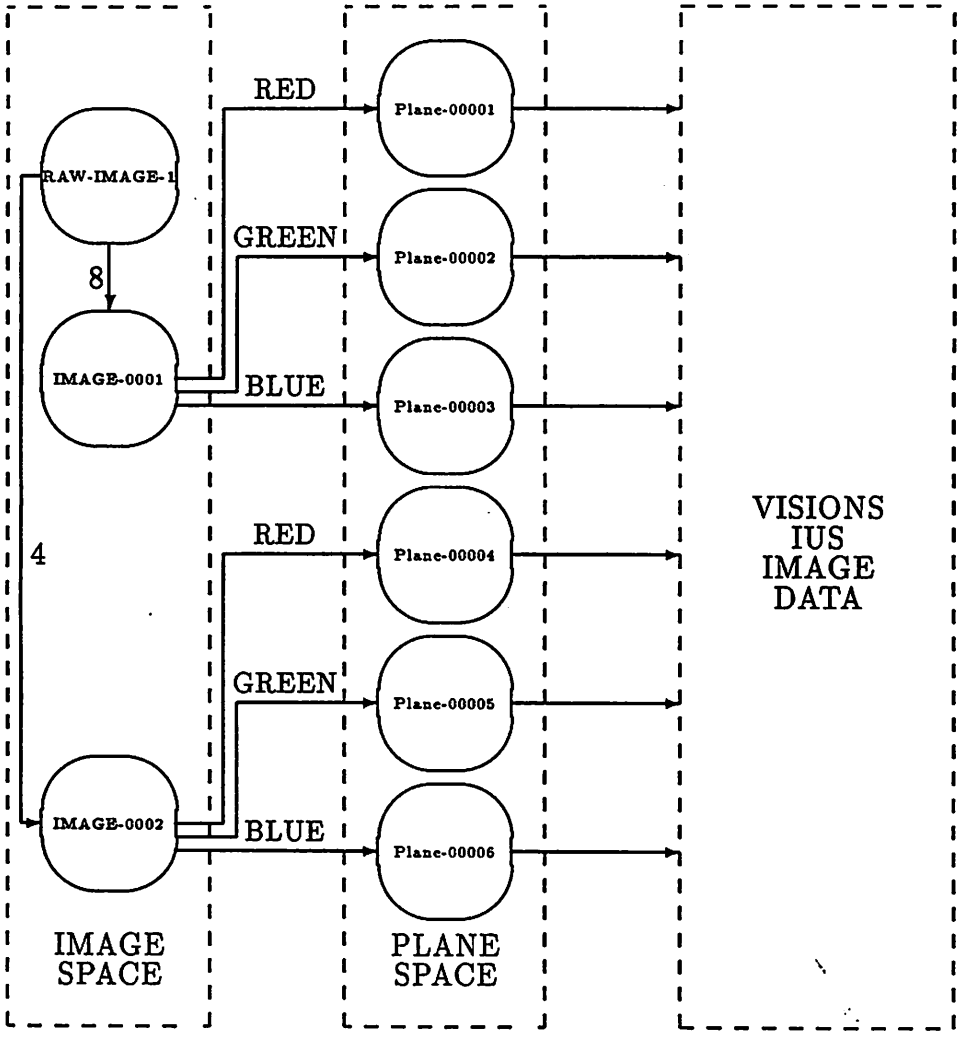


Figure 4.3: Hierarchical Image Node Structure

the system first checks for the existence of an *Image-Plane* arc attached to the Image node which has a value of the name of the desired feature. If such an arc exists, it means that the feature has been previously computed, and that the result is the Plane node attached to the arc. If no such arc exists, the Feature node corresponding to the specified feature is used to access the function necessary to compute the new feature plane which is then represented through the appropriate data structures.

4.6.2 Process Descriptions

In order to record information about the way in which intermediate–level tokens are created, certain low–level processes that occur in the system also require a representation in ISTM. Specifically, these structures are needed to record the parameters, features, and output tokens associated with Segmentation and Line-extraction processes. The process itself is represented as a node which describes specific process parameters as attributes of the value specification. Thus, in the case of a region segmentation type Plane node (Figure 4.4), we see that the value specification includes the segmentation features, segmentation algorithm, and postprocessing algorithm. The region over which this segmentation was performed is represented with the *Region-Segmentation* arc, and all region tokens produced by the segmentation process are represented by outpointing *Segmentation-Region* arcs into *region* space that have a value indicating the index of the region in the segmentation plane. The actual label plane representing the segmentation result is bound to the Segmentation node. Thus, this process node expresses not only the actual processing parameters, but is also taxonomic in that it represents information which applies to an entire class of descendant region nodes.

The representation for Line-extraction nodes (Figure 4.5) is quite similar. In this case, however, not all of the created lines are necessarily represented as nodes. Because of the large number of lines that are typically created in the

```
Plane-00016
    value= '((descriptor . "Segmentation Small Regs Supressed")
             (feature-node . Intensity)
             (file-name . "RSVSTATEDIR:ss700016")
             (isr-tokenset-name ."Rimage1$region$")
             (level . 7)
             (max-region-number . 1728)
             (min-region-number . 24)
             (plane-number . 14)
             (postprocess-algorithm . plurality))
             (segmentation-algorithm . zero-crossing))
             (time . "8-MAR-1987:22:48:59.71")
             (type . region-segmentation))
```

Figure 4.4: Region Segmentation Node Representation

```
Plane-00015
    value= '((feature-node . Intensity)
             (isr-tokenset-name . "Rimage1$line$")
             (line-extraction-algorithm . gradient-orientation))
             (level . 7)
             (max-line-index . 4331)
             (min-line-index . 1))
             (time . "8-MAR-1987:22:46.51.61")
             (type . line-extraction)
```

Figure 4.5: Line–Extraction Node Representation

line extraction process, the overhead of creating a node for each line is significant. Thus, only those lines which have been deemed "important" by the control process that initiated the line extraction process are represented as nodes. All lines are represented in the ISR, however and are identified through the tokenset attribute of the line extraction node that identifies a subset of the line tokens in the ISR.

4.6.3 Hypotheses

Hypotheses are represented in the system as individual nodes each representing a unique assertion about another entity in the system. Thus, the binding of an entity to a hypothesis is an arc that has a value expressing the belief that the assertion represented by the hypothesis applies to the entity in question. This representation was chosen over a more dynamic representation that would represent each assertion about each individual entity as an individual node. By maintaining a unique node for every assertion that can be made, we provide bidirectional access to the assertions; we can easily access all hypotheses pertaining to each entity or, conversely, we can access all entities which pertain to a particular hypothesis. As will be shown in Chapters 5 and 6, these assertions are critical to the use of knowledge at the intermediate level of processing. The assertions themselves may be quite specific (e.g. "`LINE-00054` can be considered a short, low-contrast line in this context"), or they may express higher level hypotheses (e.g. "`Region-01544` should be resegmented in this context").

4.6.4 Region Tokens

As was noted in Section 4.4, the data structures of GOLDIE were implemented prior to the introduction of the ISR in the VISIONS system. Therefore, for historical reasons, a region token at the intermediate level is represented both in the semantic network and in the ISR. This dual representation is achieved by giving the identical name to both entities. A region token is formed by first creating a new node in the semantic network, and then creating a token of the

same name in the ISR. A generic access function is used to access any information about feature values or relations of the region token, so that the region token has an external appearance of being a single entity.

In the semantic network, *region* space is the GRASPER partition that contains the set of `Region` nodes. The nodes of this space each represent a region token which has been created through a segmentation process, by a high–level interpretation process, or by a grouping process applied to two or more component regions.

No values are expressed as attributes of a region node, since all values relating to the token are stored more effectively in the ISR. Instead, the region node serves to express the relations of the region token to other nodes in ISTM. The types of relations that may be expressed between `Region` nodes in this space provide for the spatial and hierarchical constructs discussed earlier. Spatial adjacency of region structures is expressed through the use of the *Region-adjacency* arc. The value of this arc contains the length of the shared perimeter between the regions and a measure of the intensity contrast across the shared boundary. The *Region-hierarchy* arc type defines hierarchical region relations relative to the processing cone. The outpointing node of this unidirectional arc is considered to be the ancestor of the inpointing node. In this context, hierarchy refers to the fact that the image mapping of the descendant is a subset of that of the ancestor. Most typically, this situation will arise when the ancestor has been resegmented at the same or higher level of resolution. This hierarchical representation provides a number of benefits to the operation of the GOLDIE system. Ancestor relations provide information about image context; knowing the characteristics of the image area around an individual region token provides information that can be used in the evaluation or processing of that region token. The hierarchy also allows representation of multiple overlapping sets of segmentation results; a given region token may have a number of resegmentations that can be represented concurrently in the database. Finally, the hierarchy provides a computational advantage in

that it provides a spatial organization to the set of region tokens stored in the database.

Region-mapping relations indicate that the two regions contain the same set of pixels, but that the representation of the two regions exist at different levels of resolution. *Region-composition* relations are unidirectional arcs where the inpointing node is a meta-region formed by a virtual merge of existing region tokens and the outpointing node is one of the region tokens involved in the merge process. Such relations allow the measurement of features over sets of regions without performing an actual region merge.

A different set of arcs represent the relations between these `Region` nodes and nodes in other partitions of the GOLDIE system. *Region-hypothesis* arcs are used to encode relations between region nodes and hypothesis nodes, and *Region-line* arcs express spatial relationships between regions of the image and lines in the image.

The final type of relation that could be expressed in the network formalism is the set of relations between region tokens and image features. However, for the reasons explained earlier, these relations are expressed through the mechanisms of the ISR. The lexicon which defines the class of region tokens in the ISR is summarized in Table 4.1. This lexicon provides a very extensive representation for region tokens, currently providing the ability to access over 258 different features of region tokens. At system initialization, the *feat-mu*, *feat-sigma*, *feat-min*, and *feat-numpix* features are defined in the lexicon for each image feature *feat* that is represented in *Feature* space. Thus the definition of a new image feature in GOLDIE automatically provides the capability of measuring the values of that feature with respect to a region token. Note that the *feat-numpix* feature is not redundant with *numpix*. The feature *Numpix* represents the number of pixels defined in the region token while *feat-numpix* expresses the number of pixels in the region token for which the particular feature may be measured. Some of the image features used in GOLDIE (Chapter 3) are produced through a convolution

Table 4.1: ISR Region Lexicon

Feature	Description
Adjacency	Array of indices of region tokens with shared boundaries
Bitplane	Bit array representing the pixels contained in the region
Centroid-Col	Column index of region centroid
Centroid-Row	Row index of region centroid
Compactness	Ratio of region area to Minimum Bounding Circle
Extents	Coordinates of the Minimum Bounding Rectangle for the region
Level	Level of the region in the *Processing Cone*
Looseness	Ohta's [149] measure of region compactness
Numpix	Number of pixels in the region
Number	Label of region in the segmentation plane
Perimeter	Length of region perimeter
Planefit-alpha	Alpha angle of plane fit to intensity surface of the Intensity image feature over the region
Planefit-beta	Beta angle of plane fit to intensity surface of the Intensity image feature over the region
Planefit-gamma	Gamma angle of plane fit to intensity surface of the Intensity image feature over the region
Planefit-error	Least squares error in fit intensity surface of the Intensity image feature over the region
Feat–max	Maximum value of image feature *feat* in region
Feat–min	Minimum value of image feature *feat* in region
Feat–mu	Mean value of image feature *feat* in region
Feat–sigma	Standard deviation of image feature *feat* in region
Feat–numpix	Number of pixels in region for which image feature *feat* is valid
Type	Index for type of region (e.g. raw segmentation region, region token produced by a region merge, etc.)

of a different image feature (e.g. Mean). The values of these image features are sometimes unreliable (with respect to a particular region token) near the region boundary. Thus the pixel values of these image features are not used to compute feature mean, sigma, min, or max for a region token when the convolution mask used would have crossed the region boundary.

The remainder of features defined in the region lexicon are (with the exception of Bitplane) measures that are used by the various region evaluation rules in GOLDIE that were discussed in Chapter 3. The point of mentioning them here is to demonstrate the flexibility of the ISR token representation. If a different type of region feature is desired in the design of a new intermediate or high–level process, the feature may be added by adding the appropriate name and compute function to the region lexicon.

4.6.5 Line Tokens

Line tokens are represented primarily in the ISR. There is no need to create a line node unless some relation is to be expressed between the line token and some other node of ISTM (e.g. a *Region–line* relation).

The line lexicon (Table 4.2) is one which provides the essential information necessary to construct the line (endpoints and contrast), as well as other generally useful information which can be calculated from the line data (angular orientation, distance from origin, etc.). There is no actual need to store anything other than the endpoints and contrast, but the other values such as length or orientation are rather inexpensive to compute at the time of line token creation, and are quite useful during the execution of line–based segmentation.

`Line` nodes in the semantic network represent computational entities which have no direct representation in the pixel data of the image. Thus there are no direct links between a `Line` node and the `Plane` node as was the case with region tokens.

Table 4.2: ISR Line Lexicon

Feature	Description
Bitmap	Bit array representing the pixels which intersect the line
Col1	Column coordinate of endpoint1 (image coordinates)
Col2	Column coordinate of endpoint2 (image coordinates)
Contrast	Contrast across line
Extents	Extents array for bitmap of line
Length	Length of line (pixels)
Linetype	Index indicating the algorithm by which the line was created
Orientation	$arctan((Row2 - Row1), (Col2 - Col1))$
Rho	Distance from image origin to line measured along the line normal (pixels) (Hough space)
Row1	Row coordinate of endpoint1 (image coordinates)
Row2	Row coordinate of endpoint2 (image coordinates)
Theta	Counterclockwise angle between line normal and *column* axis measured at image origin (Hough space)
Note: endpoint1 and endpoint2 are chosen such that the brightest image area is to the right when traversing from endpoint1 to endpoint2	

Relations between Line nodes are expressed through the constructs *Line-adjacency* and *Line-composition*. As in the case of the Region-nodes, *Line-adjacency* records important spatial relations between lines in the image (e.g. adjacency, parallelism, or intersection). Such an observation may be the result of either a high level interpretation process, or by some low-level process that is designed to note these relations. *Line-composition* is a set relation that indicates that a form of grouping that has been performed by some high–level of intermediate–level process.

Other types of relations between line nodes are certainly possible [163], but in the work which has been done with lines in GOLDIE, this set of relations has been sufficient.

4.7 Knowledge Structures of ILTM

ILTM (Intermediate–level Long Term Memory) provides the framework for the representation of the image–independent *intermediate–level knowledge* structures of GOLDIE: schemas, object domain knowledge, knowledge of low–level algorithms, and general knowledge of image syntax. These structures form the knowledge base of the system in that they contain the knowledge which permits the intermediate–level schemas to utilize low–level processes to achieve high–level goals without explicit high–level control.

The schemas of the ILTM are the modular representation of the set of strategies that may be used in an attempt to satisfy the processing requests we call goals. Each schema is designed to serve one particular type of goal, and thus there is a one–to–one correspondence between the goals that can be recognized by the GOLDIE system and the schemas of the ILTM. The strategies that are expressed in the schemas specify the various low or intermediate–level tasks through which the goal may potentially be satisfied. With respect to their representation in ILTM, schemas exist as a set of schema nodes, each with the name of the goal to which they can respond. The value of the node contains the set of strategies, as

well as a list of known goal parameters that the schema is able to potentially utilize. The act of schema instantiation copies this structure into a schema instance representation so that the strategies may be executed. Chapter 5 will present the structure of schemas, goals, and schema instances in detail.

Another knowledge structure of the ILTM is the object domain knowledge that encodes information about the appearance of objects in an image that can be useful in the specification of segmentation tasks. This knowledge is represented as a hierarchical set of `object` nodes corresponding to specific semantic objects or classes. Associated with each node is the set of features that have been experimentally shown to best discriminate that object from all other objects in a training set of images. Each of these nodes also contains heuristic information in the form of a data–oriented (i.e. feature based) characterization of the object (e.g. textured, smooth, gradient), the segmentation sensitivity at which the object may be best extracted (high, medium, or low), and a set of rules which can be used by the intermediate–level system to provide crude hypotheses about the possible semantic identity of a particular region token. In a fully integrated system, this semantic domain knowledge used by GOLDIE would be integrated into the high-level object representation. But for experimental development, these two structures have been kept separate.

ILTM also contains the system's knowledge of image characteristics. These knowledge structures encode the heuristic information which permits the evaluation of a token, without any use of semantic information, to create a hypothesis as to whether further processing may be useful with respect to the token. This knowledge is expressed as a set of `token-class` nodes corresponding to the set of types of regions (textured, smooth, gradient) or lines (low-contrast, high-contrast, horizontal, long, etc.). Associated with each of these nodes is the set of rules (expressed as LISP functions) to be used to evaluate each of these types of token. As was discussed in Chapter 3, the results of the application of these rules are used to create data–oriented hypotheses about the tokens that indicate the degree to

which the image feature data reflects the "goodness" of the token with respect to a particular set of goals (e.g. a set of rules which are to be used to determine whether a region token which is categorized as textured should be resegmented).

Data–oriented knowledge also has to do with the utility of specific image features and algorithms in the creation of tokens with specific characteristics. Thus `token-class` nodes additionally contain experimentally derived lists of image features and segmentation algorithms for that particular class of tokens. This data may then be accessed by intermediate–level schemas which control segmentation processing.

Knowledge of the segmentation algorithms themselves is also expressed in the ILTM network. Each of the segmentation algorithms used in GOLDIE may be parameterized to control the characteristics of the resulting segmentation. Rather than encode the values for these parameters directly in the procedural code for segmentation, the range of parameter settings for each algorithm is expressed as the value of a `segmentation-process` node in ILTM. The schemas that initiate the segmentation processes therefore gain the ability to map the desired segmentation sensitivity into the range of parameter settings and thus control the characteristics of segmentation output without any direct knowledge of the way in which the parameters are utilized in the actual processing.

4.8 Control State Representation

Control of processing in GOLDIE is managed through two final partitions of the semantic network that represent the *control state* of the system. Goals and schema instantiations are represented as nodes of these networks, and arcs represent the relations between these entities [49,50,197]. The goal blackboard partition is a GRASPER space in which goal nodes, representing requests for intermediate–level processing, are created by the schema instance that requires the results of that processing. Any constraints on the request are expressed as

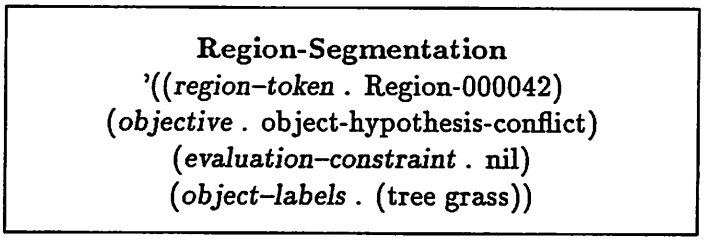


Figure 4.6: Goal Node Representation

attributes of the goal node. By using an attribute–value list to express these constraints (Figure 4.6), the schema instantiation that is posting the goal can express any information that may possibly be of use to the responding schema instance. The schema instance that is created in response to the goal will extract the value of any named attribute that may be useful in processing.

The *control process* of GOLDIE continually monitors this goal blackboard for the existence of new goal nodes. As a new goal is observed, the control process creates a new instance of the corresponding schema. The node for this schema instance is linked to the goal node by an arc that represents a contract to attempt to satisfy the goal. Any communication between invoking and invoked schema instances is achieved through a mailbox on the goal node. When the invoked schema instance has obtained results which may satisfy the goal, these results are placed in the mailbox of the goal. The invoking schema process is then able to examine the results and either accept them as satisfying the goal, or request that additional strategies be used in an attempt to produce more acceptable data. Synchronization between the contracting instances is achieved through a standard set of semaphores in the mailbox structure.

4.9 Use of the Intermediate–Level Structures in Interpretation

Before proceeding to the the description of schema–based control in the next chapter, it is perhaps instructive to give a brief example of the way in which the intermediate level representation presented here can be used to assist high–level interpretation.

A typical scenario illustrating the interaction between the intermediate–level structures of GOLDIE and the high–level interpretation system could involve a situation where a high–level interpretation process had formed two conflicting hypotheses regarding the semantic label to be assigned to a particular region token. Under the assumption that the region token actually represented an area that contained two different objects, the interpretation schema instance would post a goal (Figure 4.6) on the goal blackboard for region segmentation. The attributes of the goal node indicate the region token of interest, and additionally indicate that the reason for the goal posting was an object hypothesis conflict for the two specified object labels.

In response to this goal posting, the system would activate an instance of the region segmentation schema through the creation of a schema instance node. This schema instance would select image features and algorithms appropriate for the discrimination of the two objects (using information contained on the ILTM object nodes) and then initiate a segmentation process which would hopefully split the original region into two new regions, each representing one of the two objects. The segmentation process would access a pointer to the image feature data through the network by tracing links from the region node to be segmented, and would access the appropriate segmentation parameters from the ILTM node for the algorithm. The segmentation would be performed, and the newly created region tokens would be linked with a segmentation node in ISTM. The name of this node would be returned to the high–level process through the mailbox of the

goal node.

If the interpretation schema instance were satisfied with the results, the goal and region segmentation schema instance would be deleted. Otherwise, the schema instance would delete the segmentation node and all linked region tokens, and then perform new segmentations with different image features and algorithms until an acceptable segmentation was found.

4.10 Discussion

This chapter has demonstrated a new representation for the intermediate-level data and control structures used in the image interpretation process. These structures produce a description for data and control at the intermediate level using those techniques for knowledge representation which are normally applied only at the semantic level. This representation of knowledge has produced a twofold benefit. First, there is now a coherent representation for intermediate–level data that provides efficient bidirectional access to the data from both the high and low levels of the system. Second, as will be demonstrated in the next chapter, there is a structure through which knowledge based control may be applied to low–level processes.

It is perhaps important to stress that these data structures themselves do not serve as the mechanism by which knowledge is applied in GOLDIE. Rather, these structures provide an environment in which knowledge–based control can be used. Without data structures that are dynamic, flexible, and consistent, it would be impossible for an intermediate–level control process to interact with processes at the high and low levels of the interpretation hierarchy.

Chapter 5

Schemas and Control

5.1 Requirements for Representation of Knowledge and Control in GOLDIE

The variety of low and intermediate–level processes and data representations that may be potentially applicable during interpretation lead to a number of specific representation and control issues that must be addressed in GOLDIE. The system must support complex interactions among knowledge sources, hypotheses, processing modules, intermediate–level tokens, and image data. We require a mechanism that will allow a consistent access to each of these system levels as well as provide the capability for the representation of complex dynamic relationships between image data, image structures, hypotheses and relevant knowledge.

Because this is a research system, we also have the requirement that the control representation be flexible and extensible. In order to support a research effort, we must have the ability to experiment with new intermediate–level structures or control processes on an immediate and interactive basis.

The system should be able to function interactively with high–level processes, but it should also have the capability to independently produce a segmentation of an image that is tuned to a particular set of goals. Thus, the control mechanisms of the system should support both data-driven and hypothesis-driven processing. Focus of attention mechanisms must be included to allow the system to order the processing of the image so that we operate on the portion of the image that provides the most valuable information to the interpretation processes. Similarly, when dealing with a specific portion of the image, control mechanisms should

permit the selection of the most appropriate processing algorithms. The control structure must support the goal–directed application of segmentation and grouping operators, and the maintenance of hypotheses about the intermediate–level tokens produced by these operators.

The control mechanism must also support the parallel nature of image processing [191]. Although our processing is currently performed on serial machines, we attempt to simulate parallel operation whenever possible under the assumption that we may, at some point in the future, be able to utilize true parallel processors.

This parallelism takes two different forms in the interpretation process. The first is at the low level of processing, where most feature computation and segmentation processes can act in a spatially parallel fashion since there is no data interaction between the processes. The second form of parallelism takes place at the intermediate and high level of the interpretation process where separate interpretation processes act on different tokens of the image. In this case, the interpretation processes generally have the capability of processing with strictly local information (and therefore capable of parallel operation), but they are also capable of interacting with interpretation processes acting on spatially adjacent tokens. Since this type of interaction is necessary to maintain the global consistency of the interpretation, the system control processes must support communication structures between active instances of the interpretation processes (e.g. among active schemas). Thus, the control structures must deal with the complex synchronization issues necessary for this communication between processes at the various levels of the interpretation system.

5.2 Schema Control Mechanisms

The control for GOLDIE is specified by the mechanisms provided through the schema system that was designed by Kohl and Weymouth, initially implemented

by Weymouth [197], and later modified and re–implemented by Draper et. al. [49,51,50]. The schema bridges the gap between high and low–level systems and provides for a common form of control and communication between the various levels of the system.

As was mentioned in Chapter 4, control of processing in GOLDIE is accomplished through four different modules: goal specifications, generic schema descriptions, specific schema instances, and a goal monitor that activates the schema instances in response to the posting of goals. Goal nodes represent an explicit request for a specific type of data, and the generic schema nodes contain an expression of the parameterized set of strategies that may be used in an attempt to provide that data. As a goal is posted, representing a specific request for a specific form of data, a unique instance of the corresponding schema is created by the goal monitor as a (simulated) independent process that will attempt to provide that specific data. This newly created schema process maintains a local environment in which the specific constraints specified on the goal node define the parameters used in schema processing.

By first providing a detailed description of the structure of a generic schema, it will then become possible to show the way in which the set of active schema instances provide the heterarchical control of the entire system.

5.2.1 Schema Structures

The generic schema is a structure of ILTM which has an organization conceptually similar to the frame construct [4,131,142]. As such the basic structure of the schema is a node that has bindings for a set of named slots. These slots define the *strategies, goal, constraints*, and *local bindings* of the schema.

The *strategies* slot of the generic schema is filled by a set of procedural strategies that utilize these parameter values in an attempt to define the processes that will produce the desired data. These are independent procedural mechanisms for satisfaction of the goal; each of the strategies represents one particular way in

which the data may be produced. Depending on the specific use of synchronization mechanisms (described below) the strategies may be designed to operate in parallel, or they may execute sequentially, using some ordering that attempts to balance expected utility vs. computational cost.

The name of a generic schema indicates the specific goal that the schema is designed to serve. As a new goal is placed on the goal blackboard, the goal monitor system activates a particular instance of the corresponding schema by creating a new schema instance node that is initially an exact copy of the generic schema description. The *goal* slot is then filled with the name of the goal node that the schema is contracting to serve. Next, the constraints expressed on the goal are mapped into the local environment of the schema instance. The *constraints* slot of the instance is initially filled by an attribute/value list expressing the various types of parameters that may potentially be utilized during the execution of the schema strategies. Part of the activation procedure performs a mapping between the constraints expressed on the goal node and this *constraints* slot, overriding the default values of parameters with the values expressed in the goal constraints.

The strategies of the schema are written in a modular fashion so that several instances of a specific schema may be active at any given moment without interaction. Local state is represented by the attribute/value list contained in the *local-bindings* slot of the schema instance. This list, which is initially filled with the attribute/value pairs of constraint parameters at schema activation, provides a storage mechanism through which the individual strategies of the schema instance may dynamically create state variables to record local context. Additional local-bindings slots for data such as program counter, synchronization flags, and message buffers are created dynamically by the system at the time of schema invocation to enable the management of schema state and interschema communication protocols.

The language that is used to express each of these strategies consists of a set of standard programming constructs combined with a set of communication and

synchronization constructs. These constructs are expressed as special operators that are applied to local variables and which possess special attributes to indicate the level of interaction that the operator has with the global environment of the system. The programming constructs include the standard logical operators such as AND, OR, and NOT, and control operators such as IF-THEN-ELSE, and CASE. The system provides synchronization operators of the form WAIT-for-condition. These operators cause the schema instance to suspend until the specified event occurs.

Because of the need for parallel processing in this schema environment it is important to distinguish between those constructs that direct processes that interact with other processes of the system and those that do not. All of these programming and synchronization operators that have been mentioned to this point are defined as NON-INTERACTING because they operate only within the local environment of the schema instance; the actions of these operators have no effect on the global environment. Conversely, the communication constructs that provide the mechanisms for interaction between schema instances are all specified as INTERACTING. These constructs enable the creation, access, modification and deletion of hypotheses, goals, and sub-schema instances.

Communication between schema instances is accomplished according to two different protocols. The first is the hypothesis mechanism, whereby the schema instance asserts a hypothesis value about a token in ISTM to the system (e.g. a confidence of 0.8 that the region token Region-00284 is a *texture* type region). In GOLDIE these hypotheses are represented as values on arcs that exist between a token node (`Region-00284`) and the corresponding hypothesis node (`texture-region-hypothesis`). These arcs represent the hypothesis structure of ISTM, and are global constructs in that any process is capable of accessing or modifying the hypothesis value.

The second communication protocol is that of a contract between an invoking schema instance and the invoked schema instances. Such a contract occurs in

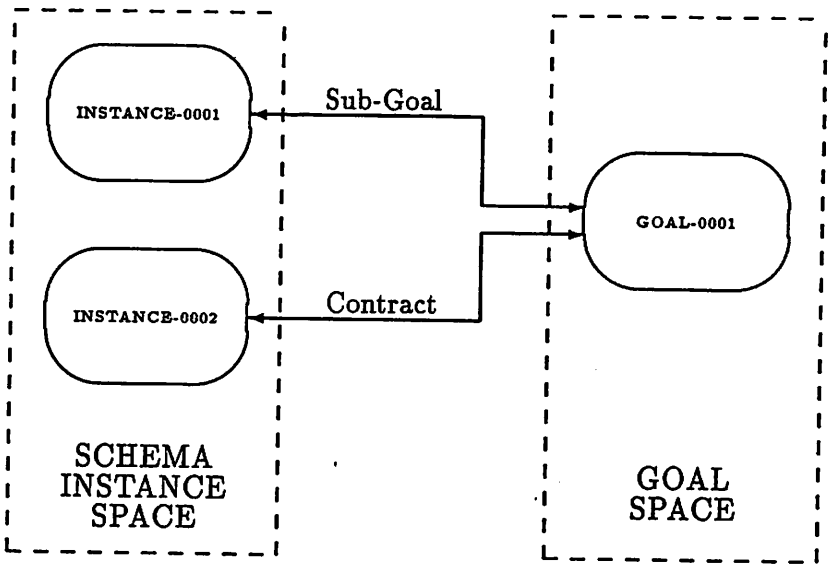


Figure 5.1: Contract Structure

response to a goal posting by a specific schema instance. The goal itself is simply a node in goal space that indicates the type of problem that needs to be solved. The goal node contains a list of all known parameters (constraints) that may be useful to a schema process which will attempt to satisfy the goal. As the new schema becomes instantiated, the two schema instances and the goal node are bound into a contract structure (Figure 5.1), and the schema instances may communicate through a message-passing mechanism that exists inside the contract structure. Example of these message passing constructs are operators such as: *contracting-instance-name?*, *respond-to-goal-request*, *wait-for-signal-to-continue*, *more-processing-possible?*, and *request-further-processing*.

There are several standard forms for the messages that can be passed by this mechanism. These include both informational and query messages. The informational messages permit the asynchronous transfer of information to the contracting instance. This allows any instance to transfer results through the contract structure and then wait for the contracting instance to read the response and take some action. Query responses, on the other hand, require a reply from

the contracting instance. Through this mechanism, an instance may request information as to whether the contracting schema instance is capable of performing any additional processing to obtain a better result. This query request suspends the querying process until the reply is received. In this manner, the message constructs provide a communication mechanism that provides local propagation of information and inter-schema synchronization.

5.2.2 Schema Operation

The control cycle for the schema system is one which simulates the parallel operation of all active schema instances. In any single cycle, each of the active schema instances is allowed to process until an operator which is defined as INTERACTING executes. At this point, processing of the particular instance is suspended for the current cycle, and the equivalent of a program counter (i.e. the pointer to the current schema instruction) is stored in the *local–bindings* slot of the schema instance. In this way, all active schema instances interact with their environment in a parallel fashion. The conclusion of a cycle is reached when all schema instances have become suspended. At this point the system state is examined to determine the status of all schema instances and goals in the system. Goals that have been satisfied are deleted (along with the contracting schema instance), and new schema instances are created to contract for new goal postings.

As the system is reinitialized for the next cycle, the "program counter" for each strategy of each of the active schema instances is examined and any schema instance that has exhausted all available strategies is deleted, with a failure message being sent to the contracting schema instance. Otherwise, each of the strategies executes until a new INTERACTING instruction is reached.

5.2.3 Evaluation Constraints

As a way of minimizing the number of unacceptable results, many schemas are capable of using a specific type of constraint known as the `evaluation-constraint`. If specified (i.e. not *nil*), this constraint is expressed as a function–value pair that is used to evaluate results prior to their being returned to the invoking schema instance. If the value returned from the application of the function to a potential set of results does not exceed the threshold value expressed in the constraint, the invoked schema instance will automatically continue processing until a set of results is found that can meet this constraint. This mechanism makes it possible to tune the schema processing to highly specific goals. For example, it is possible to create an `evaluation--constraint` such that the region segmentation schema would return only segmentations that contained regions of a certain shape or color, or both.

Although it is possible to express any arbitrary evaluation criterion with the `evaluation-constraint`, GOLDIE contains several predefined utility functions to create constraints that perform the evaluation with respect to criteria that are commonly used in interpretation processing. For example, an object–label `evaluation-constraint` may be used to determine how well a particular specified object is represented in the segmentation. If the object type is properly segmented, the semantic hypotheses for that object–label should be either very high (i.e. the region contains only the object type) or very low (i.e. the region does not contain any any portion of an object of that type) for the majority of the region tokens produced in the segmentation. In other words, there should be little ambiguity about the presence of the object in the regions of the segmentation. A parameterized utility function in GOLDIE can be used to create a particular `evaluation-constraint` for any given set of object labels.

Additional utility functions of this type exist for the creation of specific `evaluation-constraints` regarding shape and size. Shape evaluation is concerned with the compactness or rectangularity of regions produced, while size

evaluation can be used to express a preference for segmentations that contain one or more regions tokens of a specific size (expressed as a percentage of the full image).

5.3 Example of Schema Control

In Chapter 1 we presented an example that specified the general mechanism by which GOLDIE could respond to a specific high–level goal. Then, in Chapter 4, this scenario was expanded to show the way the intermediate–level data structures of the system provided the data and knowledge for this process. Now we demonstrate the schema mechanisms that provide the knowledge–based control in this GOLDIE process.

In the original example, a high level process has formed two strong object hypotheses regarding Region-00042. This region is a large region in the middle of the image, and the two object hypotheses are sky and house-wall (Figure 5.2). The existence of these two labels makes it reasonable to assume that the region is actually a segmentation error, and that the region should be resegmented into separate sky and house-wall regions. The high level system would therefore post a goal of the form shown in Figure 5.3. This goal-posting results in the instantiation of the region segmentation schema. The specification of the goal constraints `region`, `objective`, and `evaluation-constraint` provide the necessary context that enables the schema instance to start processing.

Due to the modular nature of the ISTM data structures, the specification of the name of the region token provides the newly created instance with access to all available hypothesis and intermediate–level feature data for the target region that may be necessary in the segmentation process. The `objective` attribute represents the objective that the contracting schema instance had in posting this goal, and it can therefore be used in the selection of an appropriate segmentation strategy.

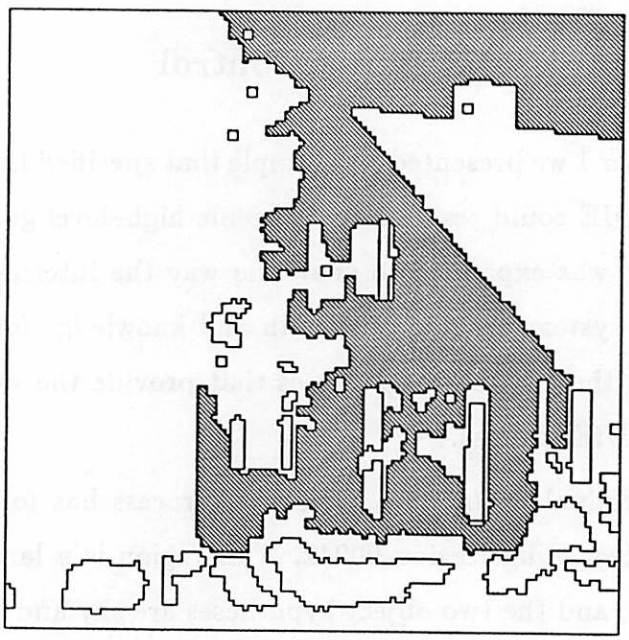


Figure 5.2: Region for Resegmentation

Region-Segmentation
'((*region–token* . Region-000042)
(*objective* . object-hypothesis-conflict)
(*evaluation–constraint* . smooth-region-constraint)
(*object–labels* . (sky house-wall))

Figure 5.3: Goal Node

The `evaluation-constraint` describes the mechanism by which the set of segmentation results may be evaluated. As the segmentations are created, they are evaluated with the LISP function described in this constraint. If the results are acceptable, the name of the segmentation node is returned to the contracting schema instance through the execution of the *respond-to-goal-request* instruction. The region segmentation schema instance then executes a *wait-for-signal-to-continue* instruction that suspends schema processing until the contracting schema instance has had an opportunity to evaluate the data. In this particular example the `evaluation-constraint` is specified as the smooth–region–constraint; this constraint requires that a large majority of the region tokens in the segmentation data have low feature variance.

If the high–level schema was satisfied the segmentation results, it would delete the goal and the contracting schema instance. Otherwise, it would send the query *more-processing-possible?* which checks a flag in the *local–bindings* slot of the contracting schema instance. Given an affirmative reply, the high–level schema would then send the message *request–further–processing* and either wait for a new reply or continue processing, depending on the nature of the high–level strategy. On receiving the request for additional processing the region segmentation instance would continue processing, and all region segmentation data that met the `evaluation-constraint` would be passed to the contracting schema instance. This processing would continue until a segmentation result was accepted or until all segmentation strategies were exhausted.

5.4 Intermediate–Level Schemas of GOLDIE

Given this general schema mechanism, it is now possible to demonstrate how schemas can be used to provide an interface between the various levels of the interpretation process. The set of intermediate–level schemas that have been implemented in the GOLDIE system are shown in Figure 5.4 and Table 5.1. Although the figure implies a hierarchy of schemas in the system, this hierarchy

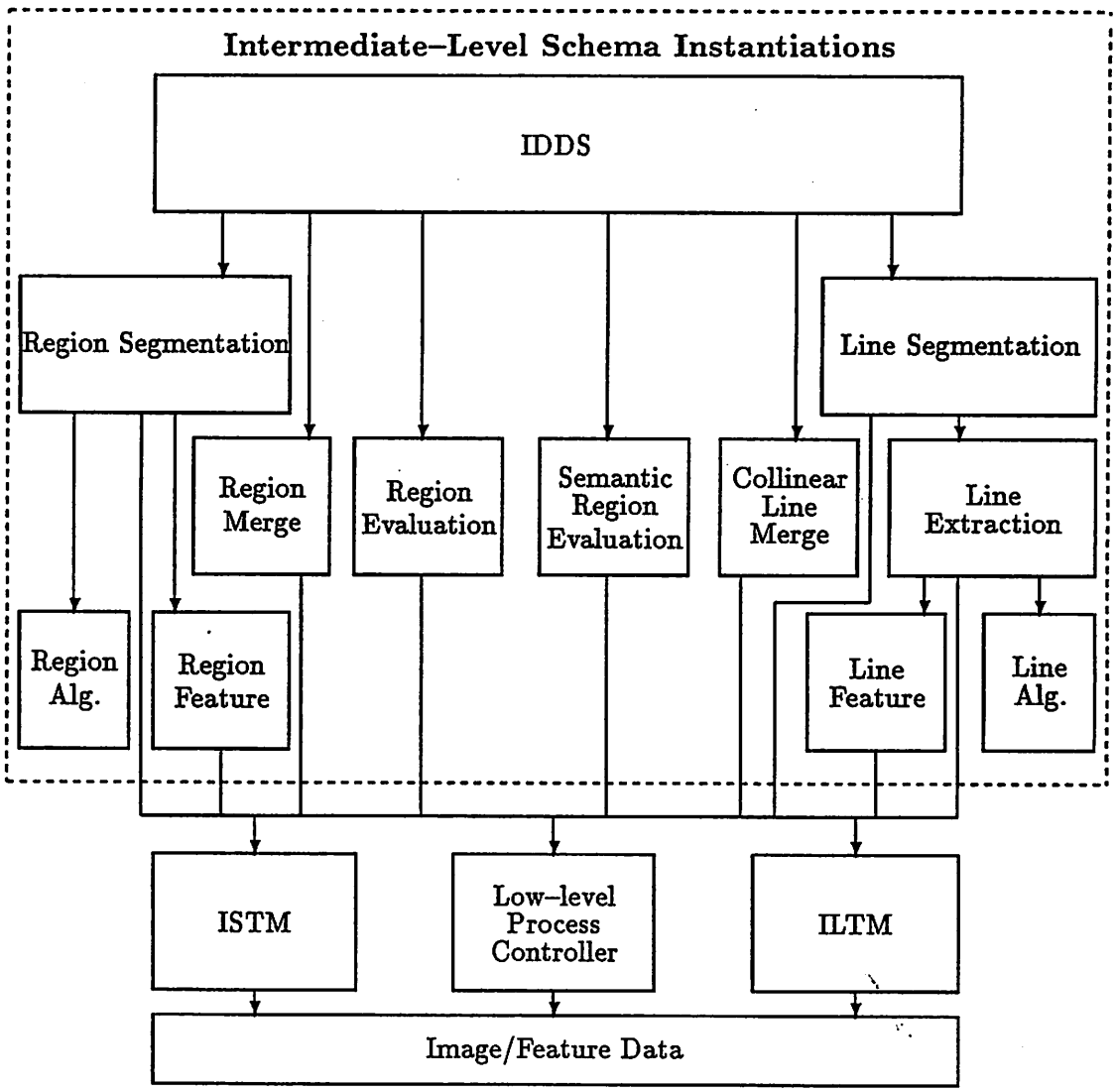


Figure 5.4: Schemas of the GOLDIE System

Table 5.1: Intermediate–Level Schema Descriptions

Schemas of the GOLDIE System	
Schema Name	Action
IDDS	Obtain the "best" segmentation of the specified region token (possibly the whole image) with respect to (possibly initial) specified constraints
region–segmentation	Segment a region token
region–algorithm	Select an algorithm for region segmentation
region–feature	Select an image feature for region segmentation
region–merge	Merge adjacent region tokens which meet criteria selected as a result of the constraints
region–evaluation	Evaluate region tokens for resegmentation or merging with respect to specified constraints
semantic–region–evaluation	Provide crude hypotheses for the semantic identity of the specified region tokens
line–segmentation	Segment region tokens using evidence from a (set of) line token(s)
line–extraction	Extract line tokens from image data
line–algorithm	Select an algorithm for line extraction
line–feature	Select an image feature for line extraction
collinear–line–merge	Merge adjacent line tokens which meet the criteria selected as a result of the constraints

is designed only to demonstrate the relationship between the IDDS schema and the other schemas of the system. We will use this hierarchy as the basis for our discussion, although it is important to recognize that the actual interaction between the schemas is more complex than indicated by Figure 5.4.

5.4.1 The IDDS Schema

At the highest level of the hierarchy is the **IDDS** (Initialization Data Directed Segmentation) schema. This schema is typically intended to be invoked at system startup to provide the initial set of image tokens for the interpretation process. Since the interpretation schemas of the high–level system are designed to operate across sets of tokens, not the raw image data, no interpretation processing may take place until we have the initial segmentation produced by the IDDS schema.

The intent of this schema is to produce the "best possible" segmentation of a region token (initially representing the entire image) without any interactive assistance from the interpretation processes. If used to provide the initial segmentation data, the only constraints on the satisfaction of the initialization goal are the *a priori* constraints of the overall system. For example, if we were implementing a system whose primary goal was to identify different types of trees in outdoor scenes, the *a priori* constraints on the goal for initialization would be "*emphasize regions with texture characteristics*" and "*emphasize tree objects*". In this way, the initial segmentation that is presented to the interpretation system has already been at least partially tuned to the overall goals of the interpretation process.

The specific goal constraints that are recognized by the IDDS schema are summarized in Table 5.2. The intent of most of these constraints should be obvious with respect to the rules and data structures presented in previous chapters. However, the somewhat obscure distinction between the two different types of constraint for both granularity and characteristic are related to the recursive nature of this schema, and will be explained below.

Table 5.2: Constraints Known to the IDDS Schema

Constraint	Description
Evaluation–criteria	Function/value pair to be used to evaluate the characteristics of the set of region tokens produced.
Object–labels	The set of object labels of interest.
Objective	The reason for invocation of the IDDS schema.
Other–constraints	An arbitrary attribute/value list that can be used to express any constraints which may be of use to any lower–level schemas.
Region–characteristic	The desired type of region token (e.g. smooth, textured, gradient). This constraint is used only with respect to this particular region token, not necessarily for recursive invocations.
Region–granularity	The desired granularity of the resulting segmentation (how closely the segmentation should respond to small variations in the image feature data e.g. low, medium, or high response). (to be used only with respect to this particular region token, not necessarily for recursive invocations)
Region–token	The region token which defines the area to be segmented.
Segmentation–characteristic	The desired type of region token to be produced by the segmentation process.
Segmentation–granularity	The desired granularity of the resulting segmentation.

The IDDS schema directly or indirectly makes use of all of the other intermediate–level schemas of the system in its initial attempt to provide the "best" data–directed segmentation, and is the most complex schema to be implemented in GOLDIE. Although the schema contains only a single strategy, the execution of this strategy involves region segmentation, region merging, line segmentation, region evaluation, and recursive invocation to achieve the desired result. Briefly, this strategy consists of the following sequence of processing steps that were discussed in Chapter 3.

1. Perform the best possible region segmentation that can be achieved according to the image features, algorithms, and evaluation functions known to the system. (*cf.* Section 3.3)

2. Perform line extraction. (*cf.* Section 3.7)

3. Insert long, high contrast lines which have been found in the raw data into the segmentation. (*cf.* Section 3.8.1)

4. Perform semantic evaluation of the new set of region tokens. (*cf.* Section 3.5.2)

5. Perform data–oriented evaluation of the new set of region tokens. (*cf.* Section 3.5.1)

6. Recursively invoke the IDDS schema over the set of region tokens that demonstrate a need for resegmentation. (*cf.* Section 5.4.1)

7. Perform region merging over the region tokens that demonstrate potential as merge candidates. (*cf.* Section 3.4)

As a result of the goal constrained evaluation steps, as well as the recursive invocations of the schema itself, the execution of this strategy is highly dynamic. Even given the same image, two different sets of goals for the IDDS schema can produce very different processing paths.

The schema requires the specification of a region token as one of the goal constraints in order to define the area over which the segmentation is to take place. In the case where the schema is being used to initialize the system, the bitmap for this region token is defined as the entire image. If this region token is correctly specified (i.e. represents a single existing region token in ISTM), the first step of the strategy is to post a goal for region segmentation on this region token (Step 1). Unless overridden by goal constraints, the IDDS schema instance will specify an `evaluation-constraint` on the region segmentation goal that makes use of the region evaluation hypotheses created by an instance of the region–evaluation schema. The LISP function specified by this this `evaluation-constraint` requires that the majority of the region tokens produced are hypothesized to be "good". In this context, "goodness" would indicate that no further segmentation was necessary, and therefore the evaluation function computes a value based in part on the resegmentation hypotheses associated with each of the region tokens in the segmentation. Obviously, if this were the only factor, the best segmentation could be one in which each pixel were represented as an individual region token, and thus other factors such as the number of region tokens and the relative sizes of the region tokens are also used in this function. The specification of this particular `evaluation-constraint` is an attempt to balance the cost of further segmentation against the cost of region merging in order to produce the highest quality segmentation at the lowest computational cost. Figure 5.5 shows the segmentation selected by the region segmentation schema according to this `evaluation-constraint` for the image from Figure 1.2.

Once a region segmentation has been obtained, it is typically the case that despite good overall quality, the segmentation data may still contain many obvious (to an observer) errors that can be corrected at the intermediate level. One such category of error is the possible absence of significant long lines that are obvious in the raw data. Therefore, the next step in the IDDS strategy is to post a line–segmentation goal for the insertion of all long, high contrast lines (Step 3). In

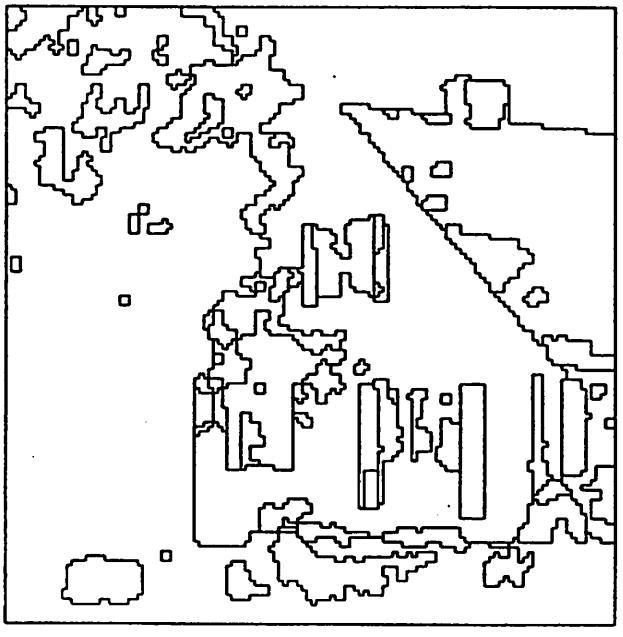


Figure 5.5: Top–level Segmentation Produced by IDDS Schema

this process, which is somewhat similar to the process employed by Nazif [140], a set of line tokens is extracted (if necessary) from the raw data by an instance of the line–extraction schema (Step 2) and the long, high contrast lines (Figure 5.6) are used to split any existing region tokens that they intersect (Figure 5.7).

The execution of this stage of the strategy attempts to assure that all significant discontinuities in the raw image data are represented by the boundaries of region tokens in the segmentation, but fails to assure that all diffuse object boundaries (i.e. those represented by slow spatial changes in feature values) will be represented. Therefore, long line insertion is followed by the posting of a goal for the region–evaluation of the region tokens (Step 6). Through the execution of the region–evaluation schema, it is expected that when a given region token spans such a diffuse boundary, the evaluation rules used by the schema will produce a strong resegmentation hypothesis. Figure 5.8 indicates, with cross–hatching, those regions for which the resegmentation hypotheses (Step 5) are

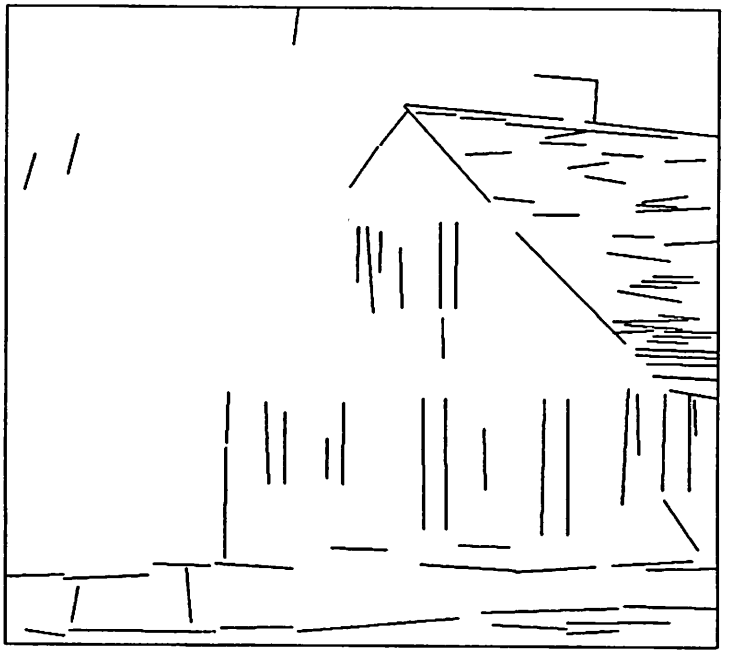


Figure 5.6: Long High Contrast Lines

strong enough to indicate a need for resegmentation (the stronger the hypothesis, the more dense the cross–hatching).

Using the goal mechanism of the system to recursively invoke the IDDS schema over the set of region tokens for which such a hypothesis exists, the segmentation may be refined in the image areas which contain these diffuse boundaries. Since the image data becomes progressively more localized as the system proceeds with this recursive invocation, more use may be made of local context to focus on the correct algorithms and features for segmentation. For example, by using semantic hypotheses produced by an instance of the semantic–region–evaluation schema (Step 4), the instances of the IDDS schema are able constrain the instances of the region segmentation schema that they invoke. Thus, if an IDDS schema were invoked over a region token that spanned the boundary between bush and tree (as indicated in Figure 5.9), the constraining knowledge that both of these semantic objects might be present in the region could help the associated region

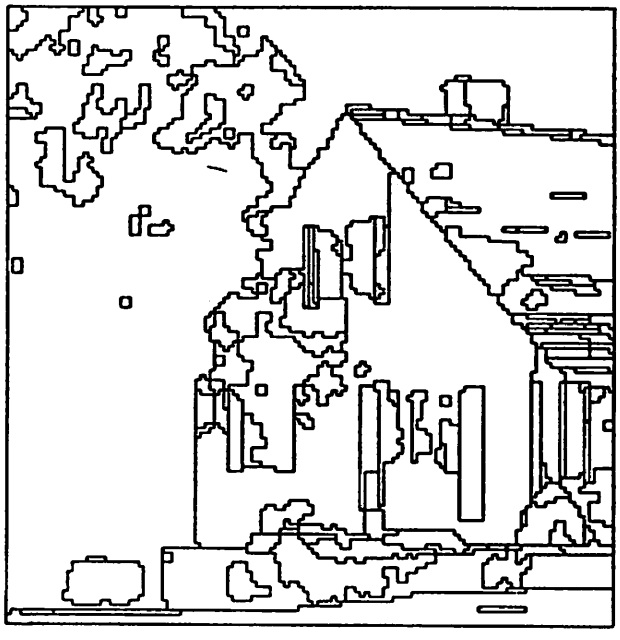


Figure 5.7: Insertion of Long High Contrast Lines

segmentation schema instance to select features and algorithms that could best discriminate these two objects.

It is for this reason that the goal constraints listed in Table 5.2 have two representations for granularity and characteristic. `Segmentation-characteristic` and `segmentation-granularity` are constraints that are to be observed at all levels of processing; they may not be overridden in any recursive invocation of the schema. If these are not specified, however, the schema is free to specify the `region-characteristic` and `region-granularity` constraints so as to make use of local context when invoking a recursive instance. Thus, not only is it possible to specify the constraints for tree and bush in `object-labels`, it is also possible for the IDDS instance to specify that the `region-characteristic` is *texture*[1] and the `region-granularity` is low.

[1]Here we are using *texture* as the intermediate-level classification described in Chapter 3.

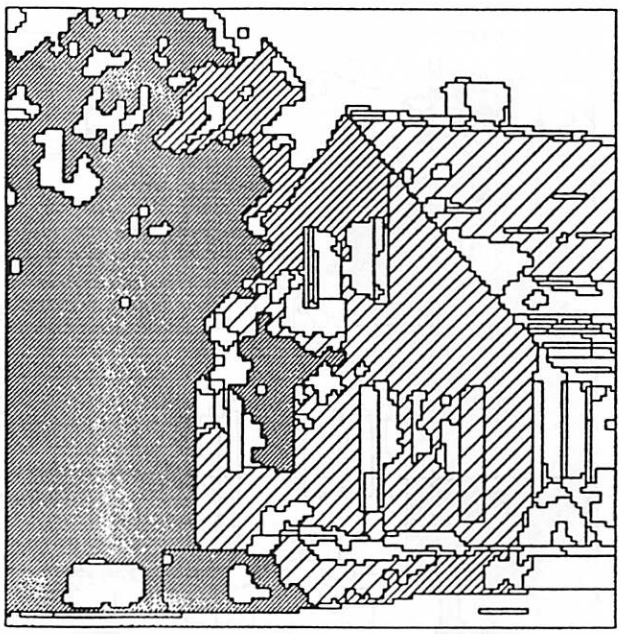


Figure 5.8: Regions with Strong Resegmentation Hypotheses

Figure 5.10 demonstrates the way in which this recursive invocation can improve the quality of the segmentation data. In this particular case, the overmerged region on the left of Figure 5.8 is resegmented using the hypotheses for the existence of both bush and tree in the region. Based on this knowledge, the image feature that is selected for the segmentation task is Objfeat–bush-tree, a rotation of RGB space that maximally distinguishes the mean RGB of bush objects from that of tree objects. With respect to the original image in Figure 1.2, it can be seen that the segmentation in Figure 5.10 has distinguished tree from bush and has captured all of the highlight/shadow boundaries in the tree. However, the segmentation is far from ideal in that the two different types of bush in the bottom left have not been distinguished, and also in the fact that many of the tree highlight/shadow boundaries are quite local and semantically unimportant. But since the area defined by the original tree/bush region (Figure 5.9) is being processed by a complete recursive instance of the IDDS schema, the remaining

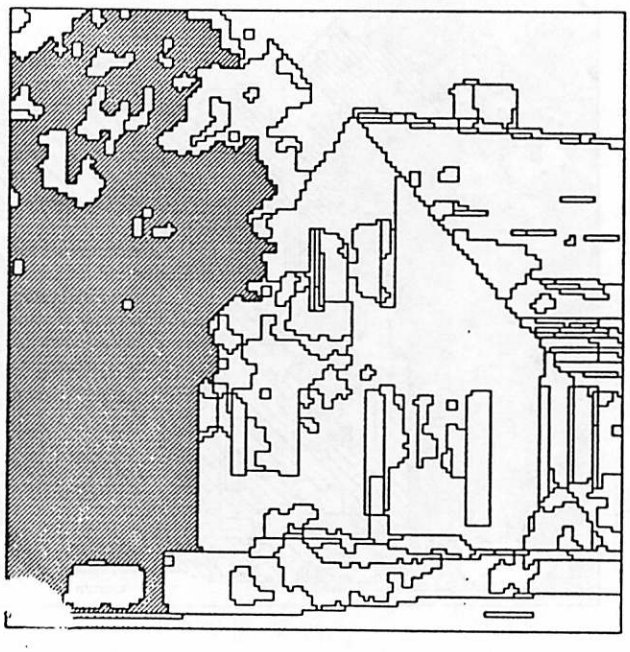


Figure 5.9: Tree/Bush Region That Has Strong Resegmentation Hypothesis

tasks of the IDDS strategy, including long, high contrast line insertion, potential recursive invocation of the IDDS schema, and region merging (see below) are used to complete the processing on this area. The final result that is returned by this instance of the IDDS schema (Figure 5.11) identifies all major semantic boundaries in the area without a great deal of fragmentation[2]. Note that several of the small regions near the top of the image that appear to be fragmentation artifacts were not defined as being part of the resegmentation region, and thus may not be merged by this particular instance of the IDDS schema.

At the point that all recursive invocations of the IDDS schema have completed, and all region tokens in the ISR have acceptably low resegmentation hypotheses, experience has shown that the updated segmentation will still typically display

[2]Note that the segmentation data produced by this recursive invocation of the IDDS schema has already been processed by the region–merge step discussed in the following paragraph.

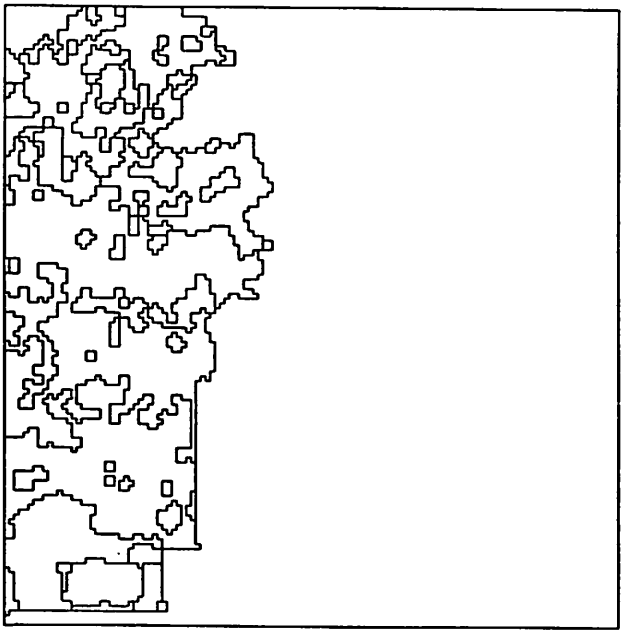


Figure 5.10: Resegmentation of Tree/Bush Region

some degree of overfragmentation (Figure 5.12). The region merge schema is therefore invoked by the IDDS schema to merge all adjacent regions that exhibit reasonable similarity under the constraints specified (Step 7).

Since the constraints specified in the original instance of the IDDS schema are passed down into any goals posted by that schema instance, this entire process occurs within the original goal context. Thus, even though a complex chain of recursive invocations of the IDDS schema has occurred, the top level constraints are observed at all levels of processing.

The final segmentation results produced by the IDDS schema (with no *a priori* constraints) on the image from Figure 1.2 are shown in Figures 5.13 and 5.14. Intuitively, the regions of this segmentation appear to provide good discrimination of the various objects in the scene, although, as has been repeatedly stressed, the only true evaluation of this segmentation would be a complete interpretation of the scene. The types of segmentation errors that do appear to be obvious (e.g.

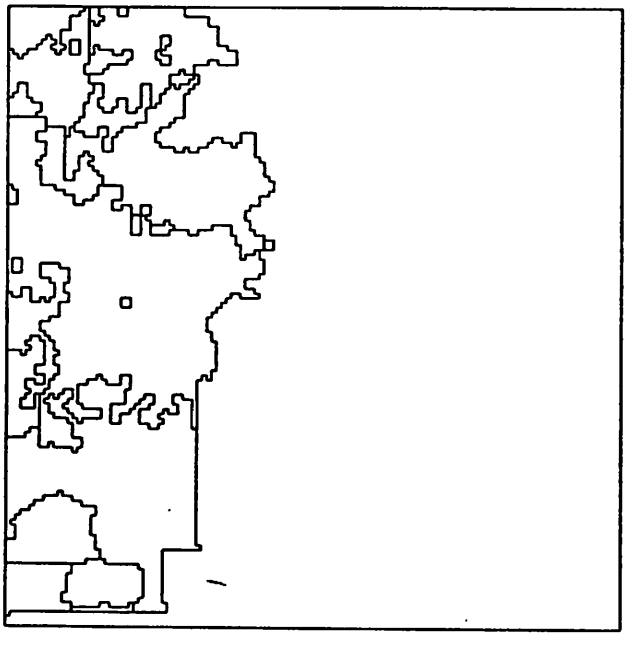


Figure 5.11: Final Result from Resegmentation of Tree/Bush Region

overfragmentation near the gutter of the house, and overmerging in the windows) are generally the result of ambiguous image data and would require high–level interpretation processing to resolve the ambiguity or direct the additional reorganization of the intermediate–level tokens.

5.4.2 The Region Schemas

The region schemas of GOLDIE (region–segmentation, region–feature, region–algorithm, region–evaluation, semantic—region–evaluation, and region–merge) provide the control for region–based processing which includes region segmentation, selection of image features and algorithms for the segmentation process, intermediate–level evaluation, and semantic evaluation of the characteristics of region tokens, and region merging.

The purpose of the **region–segmentation** schema is to produce a high

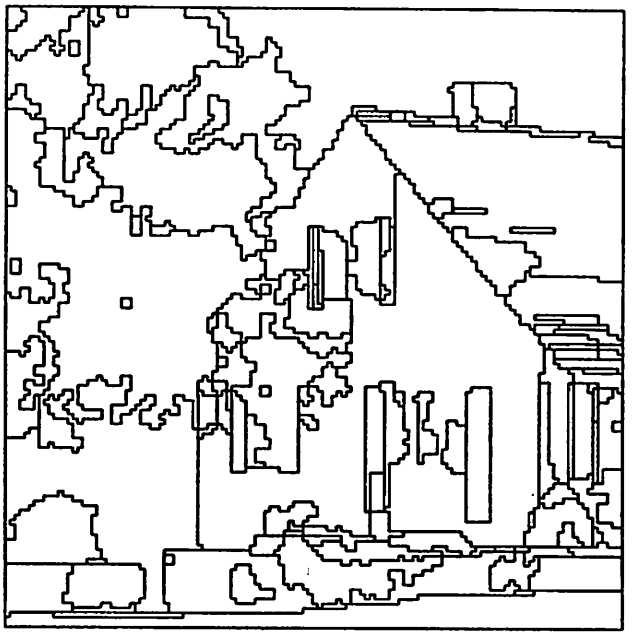


Figure 5.12: IDDS Schema Segmentation Prior to Final Merge

quality segmentation over the portion of the image specified by a region token[3]. If explicit constraints are known, such as *"concentrate on regions of the image that have the potential for being interpreted as tree"*, the schema is able to use strategies that constrain the segmentation processes such that the tokens produced should meet the expectations of the invoking higher level schema. In the case of this particular goal constraint, the schema would make use of knowledge about trees, such as the fact that they can be located in textured areas of the image and/or that they exhibit certain color characteristics, to select the low–level segmentation task that would best discriminate tree from all other objects present in the image. Table 5.3 lists the specific constraints known to this schema.

The region–segmentation schema is designed to work according to a hypothesize–

[3]Since a region token may be dynamically created by any high or low–level process to correspond to an arbitrary image area, the requirement that a region token be specified to this schema does not reduce its generality.

Table 5.3: Constraints Known to the Region–Segmentation Schema

Constraint	Description
Evaluation–criteria	Function/value pair to be used to evaluate the characteristics of the set of region tokens produced.
Object–labels	The set of object labels of interest.
Objective	The reason for invocation of the region–segmentation schema.
Other–constraints	An arbitrary attribute/value list that can be used to express any constraints that may be of use to any lower–level schemas.
Region–characteristic	The desired type of region token.
Region–granularity	The desired granularity of the resulting segmentation (e.g. low, medium, or high) (to be used only with respect to this particular region token, not necessarily for recursive invocations)
Exemplar–region–token	An existing region token that exhibits the characteristics desired in the regions created by the segmentation.
Region–token	The region token that defines the area to be segmented.

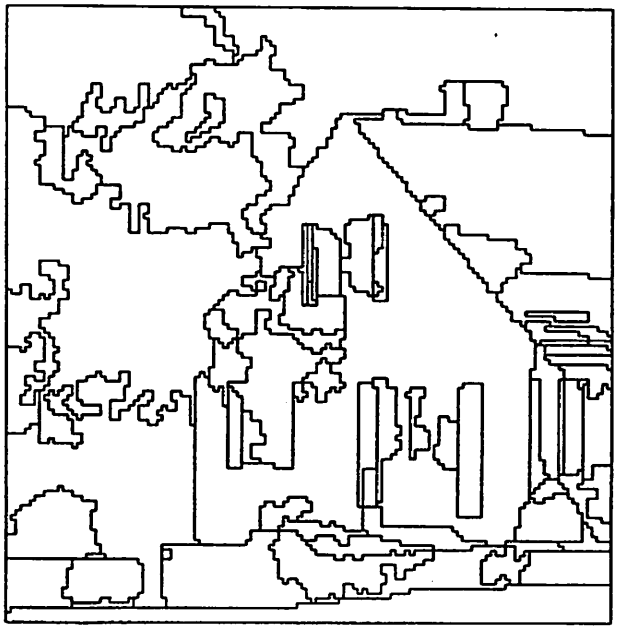


Figure 5.13: Segmentation from IDDS Schema

and–test paradigm. The schema hypothesizes appropriate processing parameters such as image features, segmentation algorithms, and sensitivity settings, and then uses these parameters to control the application of specific low–level image processing tasks through the low–level process controller. The test portion of the hypothesize–and–test paradigm is then implemented by the use of the evaluation constraint associated with the goal. If the test fails, the schema instance continues to hypothesize new features, algorithms, and settings until the test succeeds. If no segmentation is found that meets the evaluation constraint, the segmentation that came closest to meeting the constraint is returned. As an example, Figures 5.15 and 5.16 show two segmentations for the tree/bush region (Figure 5.9) that were rejected according to this mechanism prior to the acceptance of the segmentation that was shown in Figure 5.10. With respect to the default `evaluation-constraint` described above, the segmentation in Figure 5.15 received a low score due to a need for significant resegmentation, and the

Figure 5.14: Image with Overlaid Segmentation from IDDS Schema

low evaluation score for the segmentation in Figure 5.16 was due to overfragmentation. Note, however, that given a different `evaluation-constraint`, either of these segmentations could have been chosen over that of Figure 5.10.

Instances of the region segmentation schema make use of region-feature and region-algorithm schemas to establish hypotheses for the parameters involved in the specification of low–level segmentation tasks. The schema itself is mostly a control structure for the application of the hypotheses generated by these two sub–schemas. The overall strategy of the region–segmentation schema can be best expressed in pseudocode that demonstrates the loop control structure (Figure 5.17). Although this figure does not show many of the specifics of the schema (such as saving the segmentation that comes closest to meeting the evaluation criteria), it does represent the hypothesize–and–test control.

Within the **region–feature** schema, initial hypotheses for the selection of an appropriate image feature are made on the basis of the specified goal constraints

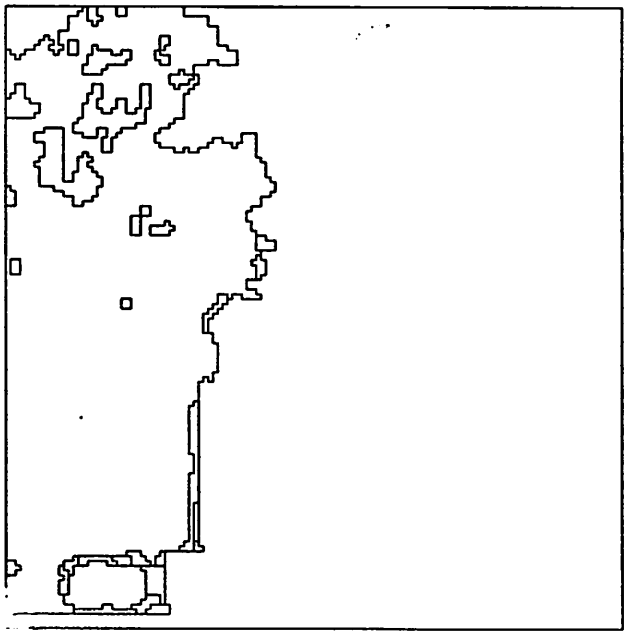


Figure 5.15: Region Segmentation that was Rejected by Region Segmentation Schema

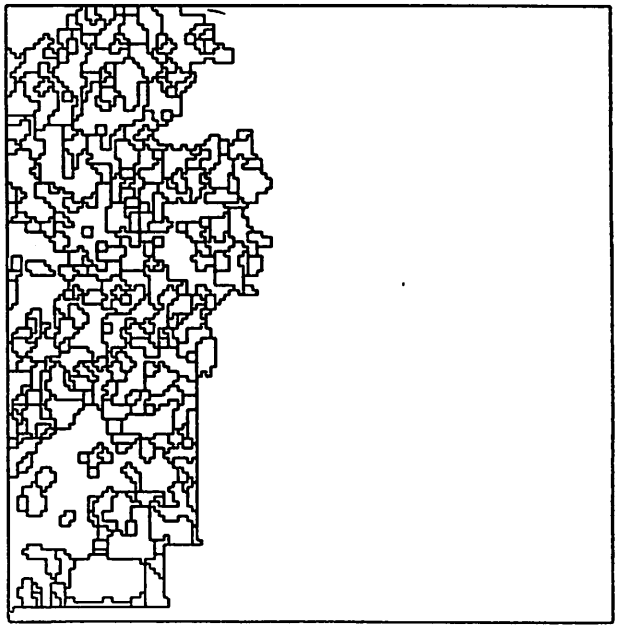


Figure 5.16: Region Segmentation that was Rejected by Region Segmentation Schema

```
post-goal(region-feature)
while (feature = get-value-from-contract(region-feature))
      post-goal(region-algorithm)
      while (algorithm = get-value-from-contract(region-algorithm))
            segmentation = apply-segmentation(region-token, feature, algorithm)
            if (meets-evaluation-constraint?(evaluation-constraint, segmentation)
                  return-value-to-contracting-instance(segmentation)
                  wait-for-signal-to-continue
            endif
      endwhile
      delete-goal(region-algorithm)
endwhile
delete-goal(region-feature)
return-value-to-contracting-instance(nil)
```

Figure 5.17: Pseudocode for Region–Segmentation Schema

(Table 5.4). Thus, if the constraints for an instance of this schema specify the `object-label` goal constraint as tree and sky, the schema would use the knowledge in the ILTM to hypothesize a set of features that would best discriminate regions representing these two objects. The region–feature schema instance would then refine these initial hypotheses by evaluating the actual frequency distribution of each of these features over the area defined by the bitmap of the region token. Based on this evaluation, the hypotheses would be ranked and returned to the invoking schema instance in order of hypothesized utility. If no element of this initial set of hypothesized image features were acceptable to the invoking schema, additional less restrictive strategies would then be employed by the region–feature schema. In this case, the schema instance would propose additional segmentation features using the additional information in ILTM that indicates the utility of specific image features for the segmentation of regions with specific characteristics,

A different strategy could be employed if the `exemplar-region-token` con-

Table 5.4: Constraints Known to the Region–Feature Schema

Constraint	Description
Object–labels	The set of object labels of interest.
Objective	The reason for invocation of the region–feature schema.
Region–characteristic	The desired type of region token.
Region–token	The region token that defines the area to be segmented.
Exemplar–region–token	An existing region token that exhibits the characteristics desired in the regions created by the segmentation.

straint were specified. In this situation, the region–feature schema would construct a potential set of image features based on the utility of those features according to the `region-characteristic` and `object-labels` constraints. This set of potential features would then be sorted according to the difference in mean values of that image feature between the exemplar region and the region to be segmented. As will be demonstrated in Chapter 6, this strategy can sometimes be quite effective in the discrimination of small objects within large homogeneous areas.

If none of the features selected according to these specialized strategies were found acceptable, or if no constraints are specified on the goal, the schema instance initially hypothesizes a set of default features that are known to typically produce good results across the particular image domain. It is this set of features that is evaluated and returned in order of hypothesized utility. This progression of more and more general strategies illustrates the design paradigm for many of the schemas of the system. The more specific the information provided in the goal specification, the more specific the data produced by the contracting schema instance.

Table 5.5: Constraints Known to the Region–Algorithm Schema

Constraint	Description
Object–labels	The set of object labels of interest.
Objective	The reason for invocation of the region–feature schema.
Region–characteristic	The desired type of region token.
Region–feature	The image feature(s) to be used in segmentation.
Region–granularity	The desired granularity for the segmentation.
Exemplar–region–token	An existing region token that exhibits the characteristics desired in the regions created by the segmentation.

The **region–algorithm** schema functions in much the same way as the region–feature schema, but the schema instance does not actually initiate any low–level processes. This schema contains a set of strategies for proposing segmentation algorithms and sensitivity settings that are based on the nature of the particular image feature and the other constraints (Table 5.5). As was described in Chapter 3, this schema is able to select between algorithms for global one–dimensional histogram clustering, localized one–dimensional histogram clustering, zero-crossings, thresholding, and global two–dimensional histogram clustering, each of which can be used with a variety of sensitivity settings. Given a particular image feature, or set of image features, an instance of this schema is designed to hypothesize the algorithm and sensitivity setting that is most likely to produce region tokens with the desired characteristics.

The schema first proposes the most specific algorithm that it can hypothesize with respect to the constraints, and if this is not accepted, it can then propose progressively more general algorithms. For example, if an instance of this schema were invoked with the constraints that the `region-feature` was Deviation (a textural feature which tends to smooth over object boundaries) and

the `object-label` was tree, the schema instance would first propose the thresholding algorithm with low sensitivity. Heuristically, the thresholding algorithm is chosen because of its ability to distinguish areas of high Deviation (tree) from area of low Deviation (non–tree) without introducing excessive fragmentation in either area. Because the one–dimensional histogram of a texture feature such as Deviation will often fail to exhibit well defined peaks, the threshold algorithm will generally give better results than any of the histogram–based algorithms. Additionally, the thresholding algorithm would be expected to place the boundary between region tokens at the midpoint of the image feature gradient of the boundary, hopefully restoring the blurred object boundary. The low parameter sensitivity would be based on the ILTM representation of *tree* that states that trees are generally best discriminated from their surroundings with segmentation processes using this setting. If the thresholding algorithm was found to be unacceptable, the schema instance would then propose a low sensitivity zero–crossing algorithm. If this were also rejected, a final hypothesis of low-sensitivity one-dimensional histogram clustering would be proposed. It is important to note that no particular knowledge about trees is represented in the region–algorithm schema itself. Rather, the schema makes use of knowledge about trees and texture that is declaratively represented on the respective nodes of ILTM.

Given certain situations, the algorithm that will be specified by this schema is fixed. For instance, if a pair of image features were specified as the `region-feature` constraint, the algorithm would have to select the global two–dimensional histogram clustering algorithm. However, the schema still has the capability to propose a number of parameter sensitivities if the initial sensitivity is found to be unacceptable. Another special case would be the situation where the `exemplar-region-token` constraint was present. In this case, it is assumed that the desired regions are small enough that their peaks would not appear in the feature histograms, so the appropriate initial choice is the thresholding algorithm with thresholds chosen on the basis of the mean and variance of the specified

Table 5.6: Constraints Known to the Region–Evaluation Schema

Constraint	Description
Region–characteristic	The desired type of region token.
Region–granularity	The desired granularity for the region tokens.
Region–shape	The desired shape for the region tokens.
Region–size	The desired size for the region tokens.
Region–tokens	The set of region tokens to evaluate

image feature across the exemplar region and the segmentation region.

Once the region segmentation schema has produced an acceptable set of region tokens, other region schemas may be used to evaluate or modify these tokens. The **region–evaluation** schema is concerned with the creation of ISTM hypotheses about region tokens that indicate a belief in the "goodness" of a particular region structure (i.e. the degree to which the existence of the region token is supported by the underlying image feature data). Based on the constraints in Table 5.6, the schema selects various subsets of the evaluation rules that were described in Chapter 3 in order to create a hypothesis as to whether or not a given region token should be merged or resegmented. The results of the application of these evaluation functions are combined, and stored in ISTM as hypotheses about the region tokens. The hypothesis values may then be accessed by other schemas of the system that will take the appropriate actions.

The schema for **semantic–region–evaluation** utilizes sets of rules that evaluate the characteristics of regions, such as short line density, color, etc. to assign a plausible semantic label to a region (e.g. sky or tree). As was stated in Chapter 3, these rules, which are stored in the ILTM, are not as precise or complete as those used by a complete interpretation system [16,15,49,50,51,80], but provide valuable information for the intermediate–level schemas in the case that no overriding hypotheses have been specified as goal constraints. Here, the only required goal

constraints specify the set of region tokens and the set of semantic labels to be considered; the schema simply applies the appropriate rules to the specified region tokens and creates the semantic hypotheses. Since this schema is designed only to produce "quick and dirty" hypotheses, the more complex evaluation strategies have been left to the high–level interpretation processes[4].

The **region–merge** schema is used by the system to reduce potential over-fragmentation. By using goal constraints to select a subset of the evaluation functions stored in ILTM, instances of this schema are able to evaluate the desirability of a merge of a set of adjacent regions, and, if necessary, direct the actual merge process. The schema may either be applied in a general fashion over the entire image, evaluating all region tokens in the ISR for merge potential, or it may be invoked to directly merge a set of adjacent region tokens that some high–level process has deemed to be similar. Again using our example of a system with the *a priori* constraint to discriminate tree and sky, two distinct instances of this schema would be instantiated in two different areas of the image. In textured areas of the image, regions would be merged if they were adjacent, at least one of the tokens had a strong hypothesis for merge potential, and they both exhibited similar textural and hue characteristics. In non–textured areas of the image, the merge criteria would still make use of adjacency and merge–hypothesis, but the regions would also have to demonstrate similar low values of feature variance as well as demonstrating co–planar intensity surface fits.

The constraints that can be recognized by this schema are presented in Table 5.7. The actual selection of which merge evaluation rules to use is made on the basis of the `region-characteristic`. The set of region tokens that are to be considered for merging is a subset of `region-tokens` in which all tokens have a

[4]Note that this is necessary to GOLDIE because the high–level interpretation schemas were not available during the development of GOLDIE. Future versions of the system in which GOLDIE and the high–level interpretation system were closely coupled could use the actual object–based interpretation schemas to create these hypotheses.

Table 5.7: Constraints Known to the Region–Merge–Schema

Constraint	Description
Boundary–merge	Flag that directs the schema to lower the merge threshold for long, thin regions between large regions.
Merge–hyp–threshold	Consider all region tokens with merge hypothesis greater than the threshold.
Merge–immediate	Flag which directs the schema to merge all specified region tokens without evaluation.
Merge–threshold	Perform merge on all region token pairs for which the merge evaluation score is greater than the threshold.
Object–labels	The set of object labels of interest.
Other–constraints	An arbitrary attribute/value list to be used to express any constraints that may be of use to any lower–level schemas.
Region–characteristic	The desired type of region token.
Region–tokens	The set of region tokens to evaluate
Semantic–merge	Flag that indicates whether to use similarity in semantic hypotheses in merge evaluation.
Texture–merge	Flag that directs the schema to consider all region tokens adjacent to large textured regions regardless of merge hypothesis.

merge hypothesis that exceeds the threshold specified in `merge-hyp-threshold` (default is 1.0). Each of these regions is then evaluated for merge potential with all of the adjacent regions. If the highest evaluation score exceeds `merge-thresh`, the merge is performed and the new region token is added to the set of potential merge candidates. The process continues until no more merges are possible.

The three flags, `semantic-merge`, `boundary-merge` and `texture-merge`, provide special types of merge evaluation that can be useful to high–level processes. For instance, once complete semantic hypotheses have been established over a set of region tokens, a high–level process may wish to base the merging process on the semantic information rather than the data–oriented evaluation. Thus an instance of the region merge schema would be instantiated with the `semantic-merge` constraint and adjacent regions could be merged based on the similarity of object label hypotheses. Similarly, once large textured areas in the image have been identified, it may be desirable to evaluate potential merges of the small regions adjacent to these textured regions and thus reduce fragmentation. Rather than forcing the high–level system to specify a merge instance for each of the large textured areas of the image, a single instance using the `texture-merge` constraint would be able to perform this operation over a set of tokens in ISTM. If the `boundary-merge` constraint is present, the merge threshold (`merge-thresh`) is lowered for long thin region tokens that lie between larger regions. In each of these cases the generic region–merge schema can become a special purpose tool for the high–level system.

This entire set of region–based schemas provide the basic set of tools necessary for the manipulation of intermediate–level region tokens. In the next section, we shall examine the set of schemas that are provided to create line tokens and how these tokens are integrated into the region representation.

Table 5.8: Constraints Known to the Line–Extraction Schema

Constraint	Description
Line–characteristic	The desired type of line token.
Region–tokens	The set of region tokens over which extraction is to be performed
Object–labels	The set of object labels of interest.

5.4.3 The Line Schemas

The line schemas (line–segmentation, collinear–line–merge, line–extraction, line–feature, and line–algorithm) are those schemas of GOLDIE that make use of line tokens to directly or indirectly modify the region tokens of ISTM.

The line tokens used by these schemas are produced, for the most part, by the **line–extraction** schema[5]. Since we have not implemented any evaluation mechanisms for line data (lines are currently used only to aid the evaluation, modification, and interpretation of region tokens), this process uses a linear control protocol rather than the hypothesize–and–test. An image feature is chosen through invocation of the line–feature schema, a line extraction algorithm is selected through the line–algorithm, and the extraction is performed. With the exception of the `region-tokens` constraint, the constraints known to this schema (Table 5.8) are not evaluated at this level, but simply passed down into the two sub–schemas.

Unlike the region–feature schema, the **line–feature** schema does not evaluate image feature data in providing an hypothesis about the most appropriate image feature for line extraction. Rather, the choice is made on the basis of the two constraints `object-labels` and `line-characteristic`. The data structures of ILTM are accessed to determine the sets of image features that best discriminate

[5]As was the case with region tokens, any high or low–level process may dynamically create a new line token that is not necessarily supported by the image data.

the desired objects and/or produce lines with the desired characteristics. The image feature that occurs most often across these sets of potential candidates is chosen as the feature most likely to produce the best set of lines.

The **line–algorithm** schema is also rather trivial in the current implementation. If all objects specified in the `object-labels` constraint are textured objects, the global one–dimensional histogram clustering algorithm is chosen. Otherwise, the gradient orientation line algorithm is specified.

Given the set of lines produced by the line–extraction schema, we still require mechanisms by which this data may be used to modify the region tokens of ISTM. As was described in Chapter 3, the **line–segmentation** schema is designed to either directly insert lines into a region representation and thereby redefine the region mapping, or to control a region segmentation process that is designed to produce region boundaries that show a high degree of overlap with a set of specified lines.

This schema is able to make use of information expressed in the constraints of Table 5.9. If the `insert-only` goal constraint is specified, processing is straightforward. The only decision required is the choice of the line extension factor, and this decision is based on the characteristics of the objects specified in `object-labels`. If this constraint is not specified, the region segmentation process proceeds according to the algorithm described in Chapter 3, in which the line specified by `line-token` is used to create two temporary region tokens representing the areas on each side of the line. A set of image features are then hypothesized, based on information expressed in `object-labels` and `region-characteristic`, and these features are then ranked according to the degree to which they discriminate between these two temporary regions.

Once a hypothesis has been established about the most effective image feature, the zero–crossing segmentation algorithm is used to perform the segmentation. The `evaluation-constraint` is then applied; if the segmentation is acceptable, the segmentation node is returned to the invoking schema process, otherwise the

Table 5.9: Constraints Known to the Line–Segmentation Schema

Constraint	Description
Evaluation–constraint	Function/value pair to be used to evaluate the characteristics of the set of region tokens produced.
Insert–only	Flag indicating that direct insertion is desired.
Line–token	The (set of) line tokens to use for segmentation.
Object–labels	The set of object labels of interest.
Other–constraints	An arbitrary attribute/value list which be used to express any constraints which may be of use to any lower–level schemas.
Region–characteristic	The desired type of region token.
Region–tokens	The set of region tokens to be segmented

next best image feature is used in an attempt to provide a more acceptable result. Processing continues until an acceptable segmentation is produced, or until all image features have been exhausted, in which case the segmentation which came closest to meeting the `evaluation-constraint` is returned.

The final GOLDIE line schema, **collinear–line–merge**, is used to find the longest possible lines that can be created from a set of line tokens through the use of the collinear line grouping algorithm described in Chapter 3. The motivation for this process may be global, where all long lines are desired, or it may be local, where a specific line is desired and the collinear grouping is performed to see if such a line can be created. In the first case, grouping is performed over the set of all existing line tokens for the image with fairly tight grouping criteria. In the second case, the grouping is performed over a small set of line tokens with weaker grouping criteria.

According to the set of goal constraints expressed in Table 5.10, the schema selects the appropriate control parameters to direct the actual grouping process.

Table 5.10: Constraints Known to the Collinear–Line–Merge Schema

Constraint	Description
Line–characteristic	The desired type of line token.
Line–tokens	The set of line tokens to be grouped
Object–labels	The set of object labels of interest.

The initial set of line tokens to be considered for grouping is specified by the constraint `line-tokens`; if none are specified, all line tokens in the ISR are potential candidates. `Line-characteristic` is then used to prune the candidate set by eliminating lines according to line contrast. If the desired characteristic is high contrast, only the high contrast lines from the candidate set will be used in the grouping process, and the grouping criteria are such that the distance between linked endpoints can be reasonably high. If, however, the constraint specifies low contrast, no lines are eliminated on the basis of contrast, but the distance between endpoints of linked lines must be small. In the absence of a specification of the `line-characteristic` constraint, the `object-label` constraint is used to set grouping criteria in accord with the type of line expected to be found associated with the specified objects.

If the set of line tokens is specified however, `line-characteristic` is not used, and grouping criteria are set according to the type of line expected in the scene representation of the objects named in `object-labels`.

5.5 Discussion

Before discussing the implications of the representation for schemas and control, it is important to emphasize the comments on the generality of the GOLDIE schemas that were made at the beginning of this chapter. We make no claim that any of these schema represent the "best" strategy for the control of the various low or intermediate–level processes. The intention of the GOLDIE system is to

provide a representation and mechanism through which the control itself may be achieved. Although each of these schemas could easily be extended to be more general and robust, the contribution of GOLDIE is that we now have a framework within which a variety of domain or algorithm experts may perform this extension in a straightforward and incremental fashion.

Schemas are tools for knowledge–based control, and the issue of control in knowledge-based systems is currently a very dynamic topic in the field of AI [23,41,42,44,51,49,60,80,129,131,181,187,195,202,204]. In view of this variety of representations, it is useful to examine the nature of the control used in GOLDIE in the context of some of the more successful systems.

5.5.1 Knowledge representation

Knowledge representation for GOLDIE is implemented primarily through the use of semantic networks, although several modifications have been made to extend the functionality of the representation in response to the special needs of this complex domain.

The representation of Intermediate Long Term Memory is an implementation of the semantic net structure that is used as the knowledge base for the high-level interpretation system. This net incorporates a hierarchical representation, including class/subclass relations and inheritance. As such the net is an example of the classic semantic net paradigm [23,94]. The network contains the system's static information regarding the nature of visual scenes: the relationship between classes of semantic labels, the visual representation of objects possessing these labels, and the characteristics of the various classes of tokens that have been represented in the system.

A second class of static knowledge representation in the system is the encoding of image feature calculation procedures in the feature space of ILTM as described in Chapter 4. This representation uses the semantic net structure to represent the active values paradigm that is usually considered to be an attribute of active

slots in a frames representation [60]. In GOLDIE, this procedural knowledge has been encoded as an attribute of nodes in ILTM for consistency with the rest of the representation. In this way, the modularity of the system has been preserved, and the operation of the system becomes independent of the set of image features selected.

A third representation is the network representation for Intermediate Short Term Memory in the **region** and **line** spaces of the network[6]. This structure represents the current state of image structures that have been constructed over the image in relation to their hypothesized interpretation state. This information can be considered the set of facts that represent the current state of the GOLDIE system. Such a representation requires structures that can encode identity, unary attributes, spatial relations, hierarchy relationships, and hypothesis relationships. The concept of nodes linked by directional arcs again satisfies this need. In this case, however, we do not have the same type of taxonomic inheritance relations exhibited in the ILTM. Although the system does include ancestry relations, these relations do not describe true class/subclass membership. In this respect, the ISTM follows the structure developed for the Blackboard of Hearsay-II [56]. Image entities are posted in this structure as they are developed, with facts encoded as attributes of the node structure, and contextual information represented through arcs to other entities in the structure.

Considered as a unit, these three network structures can be seen to encode the facts of the system. Although conceptual relations vary between the three structures, the underlying representation of nodes and arcs remains constant. By partitioning different types of nodes into different spaces of the GRASPER system, we retain conceptual and descriptive clarity while maintaining the ability to represent any possible association between entities in any of the knowledge structures.

---

[6]Future implementations of GOLDIE can and will provide a representation for additional types of intermediate–level tokens such as surfaces or motion fields.

5.5.2 Control

Although this same node and arc network representation is maintained in the control structures of the GOLDIE system, the actual mechanism bears little similarity to any previously established control formalisms. As has been repeatedly stated, the control of segmentation processing is highly heuristic; the state of the art (and most probably, the nature of the problem) precludes an *a priori* definition of a "correct" segmentation. The value of a segmentation can be understood only in relation to the system goals and resulting interpretation of the segmentation. Rather than providing a search of a discrete state space, the control mechanism for image segmentation and interpretation must create and manipulate a hierarchical set of image abstractions so as to maximally satisfy the set of interpretation goals. We have therefore developed a goal directed strategy for GOLDIE that allows heuristic hypothesize–and–test procedures to be invoked in response to explicit goals.

5.5.2.1 Production–Rule Systems

There are, however, components of this control structure that bear similarity to other AI systems. Most noticeable is the relationship between the response to explicit goals in the GOLDIE system and the backward chaining control formalism of some production rule systems, especially MYCIN [180] . This backward chaining mechanism essentially proposes the antecendents of productions as goals to the system. In this manner the system can be driven in terms of decreasing grain size in the knowledge domain. This concept has been significantly enhanced in the GOLDIE system. Our goals are proposed explicitly: they are not built into the control structure of the system, and we are therefore not restricted to production rule formalisms in the expression of our goal/subgoal relations. Schemas may define a number of strategies that can be used to satisfy a particular goal, and they also provide the control structures that permit a particular schema in-

stance to control and synchronize these strategies to provide efficient exploration of the (possibly large) set of potential subgoals. The procedural encoding of the strategies allows us to express the conditions under which a subgoal should be explored, and the inter–schema communication allows control over the amount of processing that is dedicated to the exploration of subgoals.

Through the use of explicit goals to control processing, we also provide for multichannel communication between the levels of the system. Thus, a high level interpretation process can communicate with all levels of the GOLDIE system. Any sufficiently knowledgeable schema instance may propose a goal at the very lowest levels of the system (and vice–versa). This means that although there may be a conceptual hierarchy of generality vs. specificity in the level of knowledge in the various schemas of the system, there is no inherent need to traverse all intermediate levels in an attempt to satisfy a problem.

Through this procedural encoding of the goal specification, we also avoid the global proliferation of metarules that often tend to overwhelm the design of production systems. In effect, the metarules are encoded directly within the schema procedures. This limits the scope of the metarules, and maintains locality of effect.

5.5.2.2 Means–End Analysis

It is also possible to view this system in terms of the paradigm of means/end analysis as exemplified by GPS [57]. The response to the posting of a goal is the activation of a schema which serves to provide a resolution between the current and desired states. The contracting schema instance executes actions that attempt to solve the problem specified in the goal structure, and thus transform the state. The analogy breaks down, however, when we note that we are not dealing with closed system states. Since a schema may create new image structures, such as regions or lines, which effectively become state variables, the the set of states is not *a priori* fixed. The desired result of a goal posting is not specified

as a rigorous state transition, but rather as the creation of a new state in which certain conditions hold. Thus, although the goal–directed control of GOLDIE is similar in spirit to that of GPS, the mechanisms themselves are quite different.

5.5.2.3 Frame–Based Systems

A final analogy of this control procedure is to the use of frames in expert systems [60]. As has been mentioned in Section 5.5.1, the basic structure of a schema is similar to that of a frame. We have incorporated the concept of active knowledge and functional slot values that provide much of the problem solving capability in these systems, but have found that we require a dynamism that is quite difficult to achieve in generic frame-based systems. Control in a frame–based system is produced through the propagation of changes in slot bindings. Through this mechanism, classification and state description can be performed effectively, but it is difficult to implement the mechanisms for sequencing and scheduling that are necessary for a hypothesize–and–test protocol. Image segmentation (and interpretation) require an dynamic interaction between a variety of image abstractions (pixels, regions, lines, surfaces) that would be extremely awkward under this general protocol.

5.5.3 Conclusion

In this chapter, we have described the structures and mechanisms for control in the GOLDIE system. The representational structure for control knowledge in schemas has been presented in some detail and compared to other state–of–the–art AI systems. The actual procedural control knowledge that is embedded in the individual intermediate–level schemas of GOLDIE has also been presented in an attempt to show the way these schemas are able to interact in response to specific goals in the image interpretation process. As will be demonstrated in the next chapter, the actual implementation of this set of schemas in GOLDIE allows

the system to behave nicely over a variety of image data and goals. The significance of this chapter, however, lies in the schema control paradigm. Within this paradigm, modular forms of interaction between the various levels of processing produce a dynamic system that can adjust to the overall goals of interpretation. GOLDIE is a system in which heuristic or empirical knowledge about low–level and intermediate–level tasks can be be expressed and used to control the segmentation/interpretation loop.

Chapter 6

Experimental Demonstration of GOLDIE

Although the preceding chapters have presented the structure and motivation for GOLDIE, demonstrations of actual processing are crucial to an understanding of the value of intermediate level control. The experiments presented in this chapter demonstrate the way in which the schemas of GOLDIE can respond to a variety of goals to produce intermediate level image tokens that are appropriate to the needs of interpretation processes.

6.1 Introduction

As was pointed out in Chapter 5, the schema based control paradigm can be used either in a data–directed or in an expectation–directed mode. The data–directed (bottom–up) mode is exemplified through the behavior of the the IDDS schema. This schema uses intermediate level goals and knowledge to create a set of region tokens when the high–level system has little information about the semantic content of a particular image area (for example, when a new image is acquired). The expectation–driven (top–down) mode is used when the higher level processes require the creation of a specific set of tokens with highly constrained characteristics.

In the next two sections, we will describe a variety of experiments that demonstrate the behavior of GOLDIE in the data–directed and expectation–directed modes.

6.2 Data–Directed Processing

One of the primary uses for data–directed processing is in the production of the initial segmentation of the entire image. In such situations, we can view the goal constraints as representing the general *a priori* interpretation goals, rather than expressing the need for specific interpretation–driven processing. For example, if the interpretation system were interested primarily in the identification of a blacktop road surface (as in the case of an autonomous vehicle), the overall goals for the initial segmentation would be to identify large homogeneous areas of the image and to ignore large textured areas. If, on the other hand, the intent of the interpretation processes was to identify trees and shrubs in the image, the goals for the initial segmentation process would specify a need to identify areas of the image which were uniform in texture, while spending little time in the identification of the precise boundaries of large homogeneous areas of the image.

In either case, the initial segmentation for an image should partition the image into a reasonable number of region tokens that allow the higher level processes to form initial hypotheses about scene content. If there are too few regions in the initial segmentation, important semantic cues may be missed; if there are too many regions, large semantic cues in the image may be overfragmented to the point where the large number of separate small regions prevents the higher–level processes from recognizing the attributes of the object as a whole.

Thus, the overriding goal for the initial segmentation process is to produce a set of region tokens which closely describe the set of image events that are significant with respect to the overall goals of the interpretation.

6.2.1 Initial Segmentations

In the most general case, the default function of the interpretation process is to identify all significant objects in the scene. Since all distinct image events are therefore of equal importance, the initial goal specification to the IDDS schema

would contain only the single constraint which specifies the image to be segmented (actually a region token which covered the entire image).

A variety of images have been processed under the IDDS schema of GOLDIE to produce initial segmentations that are constrained only in the sense that the resulting set of image tokens are to provide the best (with respect to the `evaluation-constraint`) intermediate–level estimate of the actual events present in the image. By comparing these results to the segmentations produced by several other purely low–level segmentation processes that have been developed within the VISIONS environment, it becomes possible to demonstrate the utility of the intermediate–level control paradigm. These examples show the way in which intermediate–level control can be used to produce segmentation data that reflects the local characteristics of the corresponding image data, thereby improving the overall quality of the intermediate–level tokens.

The specific algorithms that will be used in these comparisons were discussed in Chapter 3. The first is the region merge segmentation developed by Griffith [18], that starts with a highly overfragmented initial segmentation and then uses a set of predefined rules to iteratively merge regions until no new merges can be made. The second algorithm is the localized one–dimensional histogram clustering algorithm developed by Nagin [137] and modified by Kohler [115] and Beveridge [18]. In the examples below, these two algorithms have each been applied using the parameters and preprocessing steps that were used by the original developers of the algorithms. The intent here is not to provide a rigorous comparison of the different algorithms, but rather to show the overall characteristics of the data they produce.

Figure 6.1 shows a typical ($256x256$) RGB outdoor scene. Using the Griffith region merging algorithm, the Intensity image feature for this image is first oversegmented by a conservative region growing process to produce the segmentation shown in Figure 6.2. In itself, this segmentation is practically useless for interpretation purposes. The segmentation contains 3,599 regions, and all textured

areas of the image are quite fragmented. However, this inexpensive low–level process has served to isolate the three large homogeneous areas of the image corresponding to sky, house roof, and sunlit grass, as well as other reasonably large homogeneous areas in the house wall. Even this simple algorithm has reduced the number of data elements from 65536 (256^2 pixels) to 3599. Once the region merge processing has been performed (Figure 6.3), the total number of region tokens has been reduced to 400. In many ways, this can be considered a very good segmentation. Almost all of the significant semantic objects in the scene have been discriminated, and practically all of the important object boundaries have been preserved throughout the merging process. However, several areas of the image that are semantically homogeneous (e.g. shadowed grass at the bottom of the image, bushes, and bottom edge of the roof) are still highly fragmented. Because this is a global algorithm, the merge rules must be conservative enough that they can be applied to any pair of adjacent regions in the segmentation. Since the same set of rules is used to evaluate the potential merge of two dark, textured bush regions as is used in the evaluation of a merge of two bright sky regions, these rules must be designed conservatively enough that they will not perform an undesirable merge in either situation. Consequently, the merging thresholds are set high enough that many merges that would seem to be obvious to a human observer are not performed.

When this same image is segmented using the localized one–dimensional clustering algorithm, the image is first divided into a number of small (typically $16x16$) subimages, and then each of these subimages is segmented independently with a one–dimensional histogram clustering algorithm. Figure 6.4 shows the result of this partitioned segmentation processing using the Intensity image feature that has been enhanced using the edge–preserving Overton–Weymouth operator [151]. Using the merge criteria developed by Kohler, region merging is performed across the artificial sector boundaries, resulting in the segmentation shown in Figure 6.5. Obviously, the set of region tokens produced from this segmentation

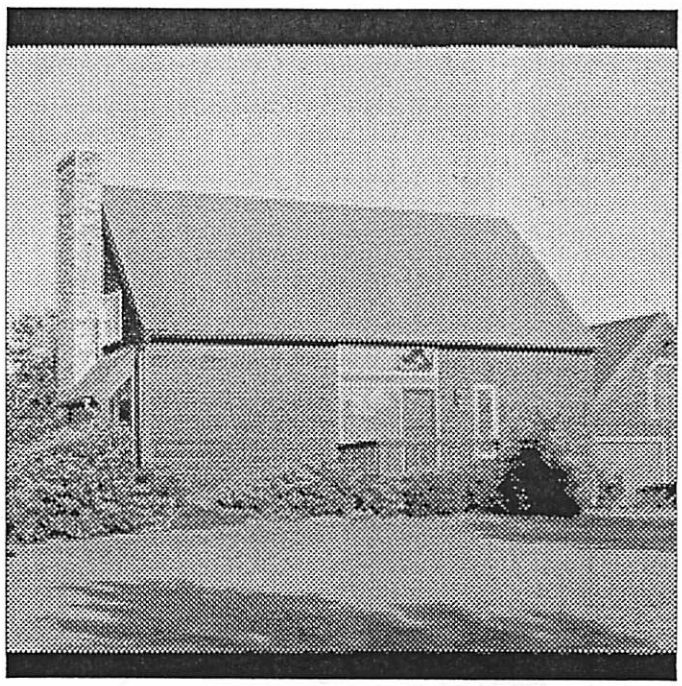


Figure 6.1: Outdoor House Scene (House 15)

is completely inadequate for interpretation purposes. With the exception of the house roof and the sunlit area of grass, none of the important object boundaries are present in the segmentation. If, however, we adjust the sensitivity of the algorithm using the one parameter setting defined by Beveridge (very–low, low, medium, high, and very–high)[1] [18], the characteristics of the segmentation can vary significantly (Figures 6.6–6.9).

[1]These five parameter settings of this parameter correspond to the five different sensitivity levels defined for this algorithm when used in GOLDIE

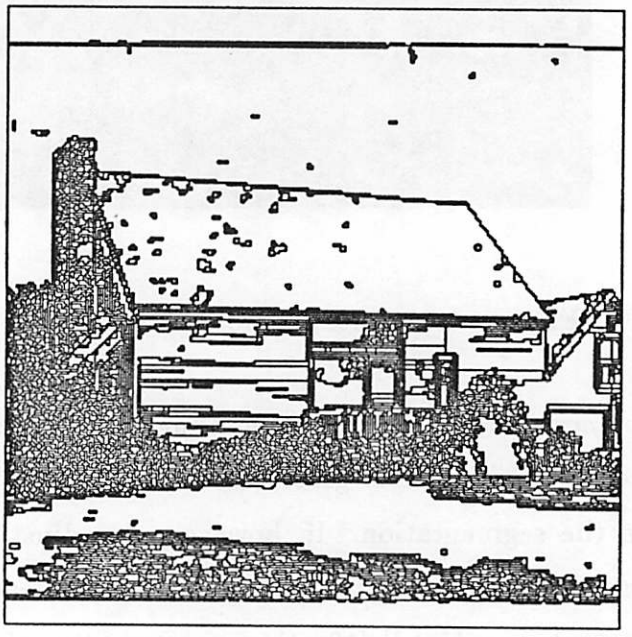


Figure 6.2: Oversegmentation of House 15

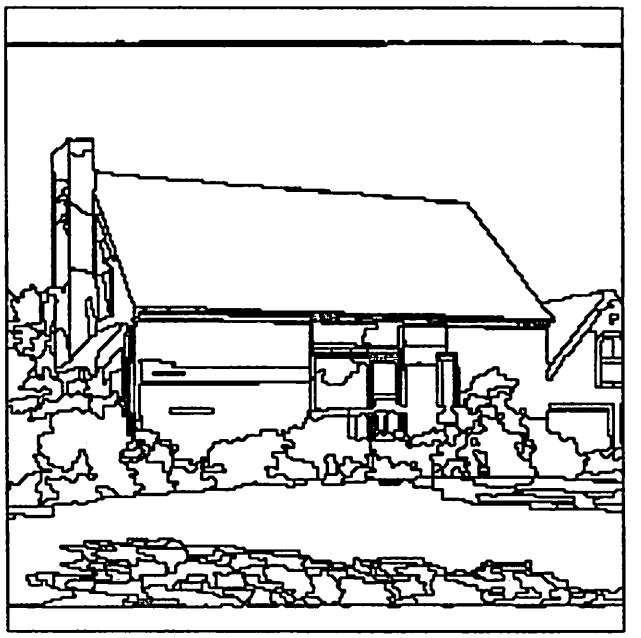


Figure 6.3: Region–Merge Segmentation of House 15

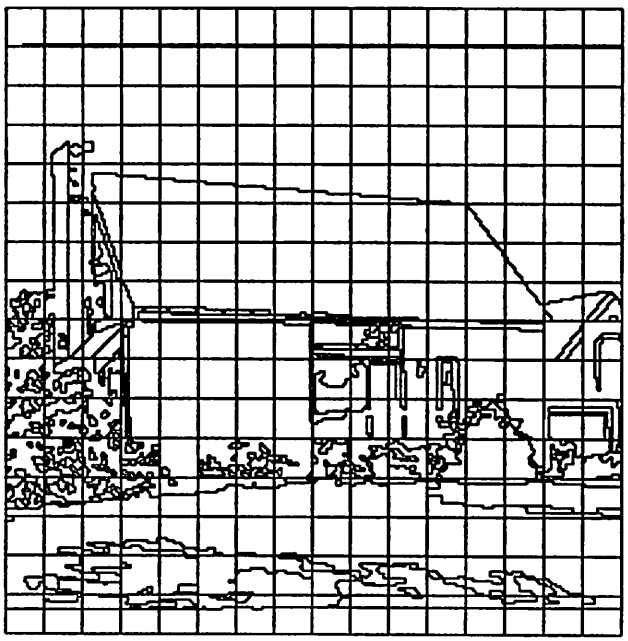


Figure 6.4: Partitioned One–Dimensional Segmentation of House 15 (Medium)

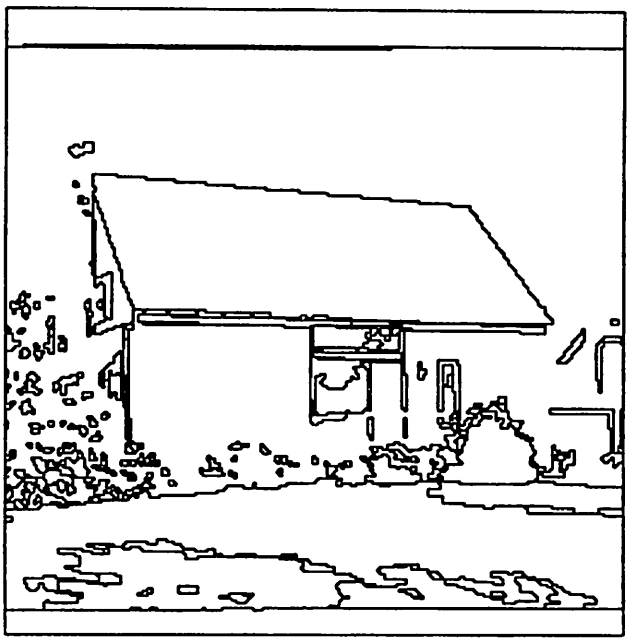


Figure 6.5: Localized One–Dimensional Segmentation of House 15 (Medium)

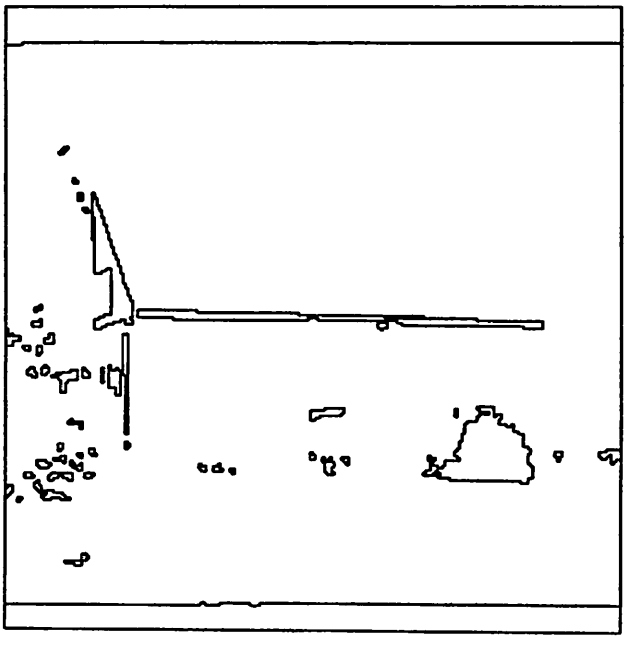


Figure 6.6: Localized One–Dimensional Segmentation of House 15 (Very–Low)

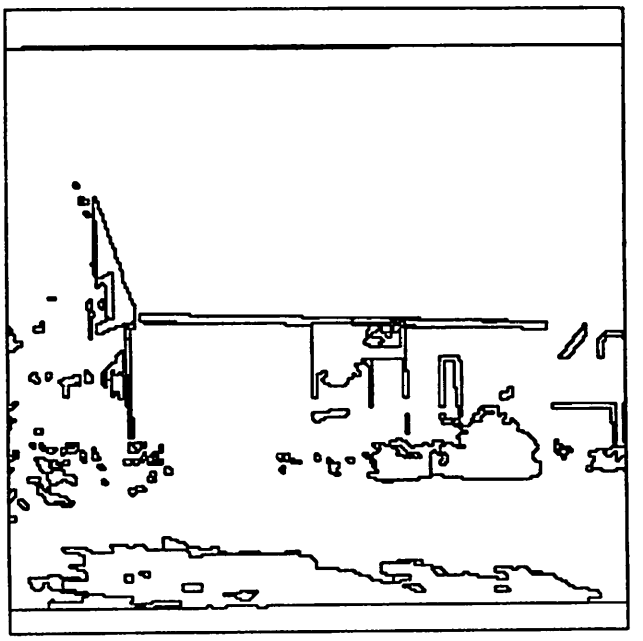


Figure 6.7: Localized One–Dimensional Segmentation of House 15 (Low)

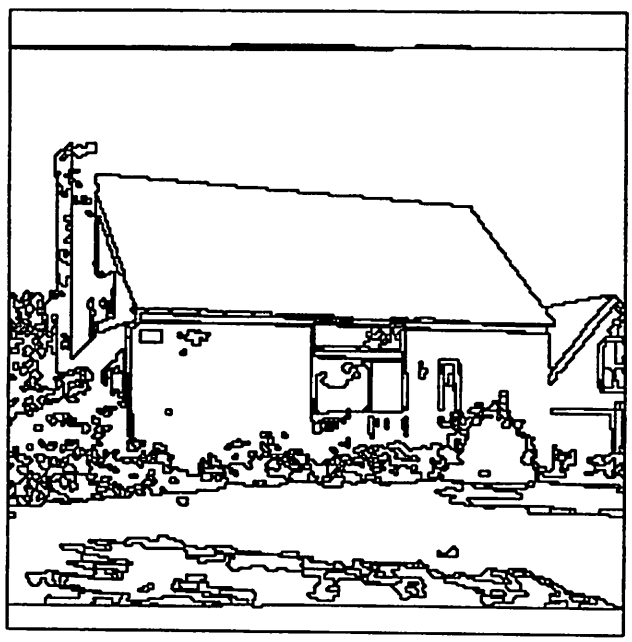


Figure 6.8: Localized One–Dimensional Segmentation of House 15 (High)

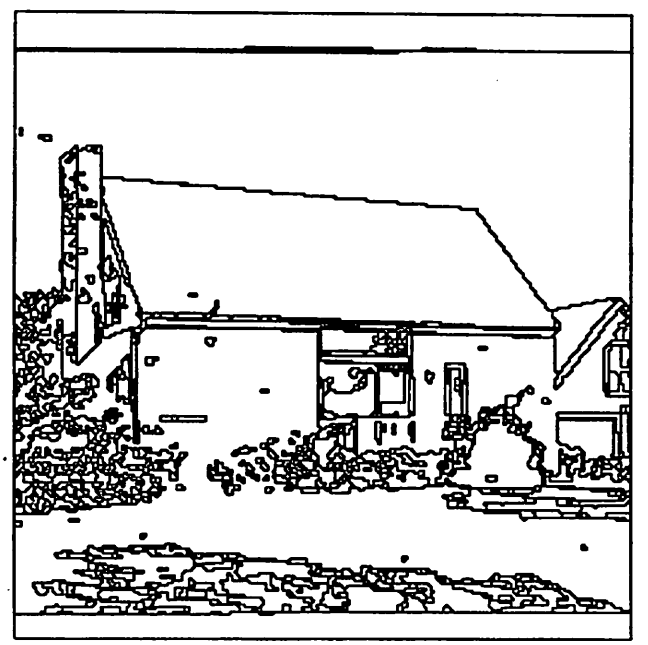


Figure 6.9: Localized One–Dimensional Segmentation of House 15 (Very–High)

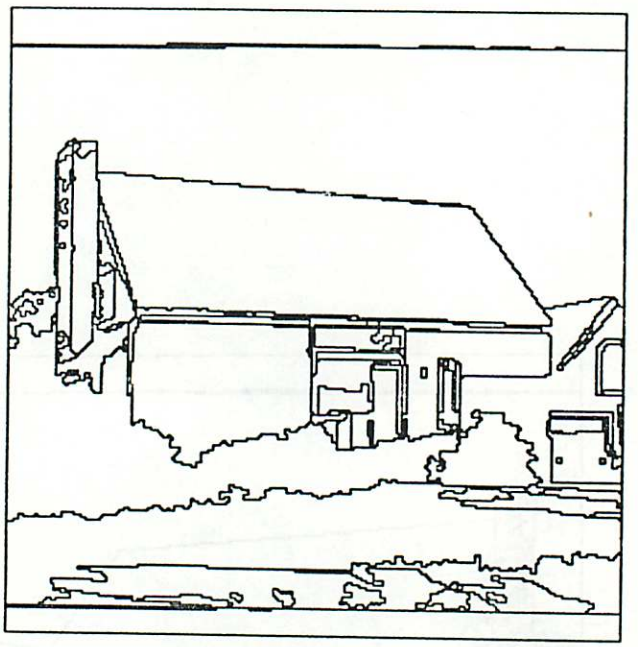


Figure 6.10: GOLDIE IDDS Schema Segmentation of House 15

Figures 6.13 through 6.16 compare these three segmentation processes on a different house image (Figure 6.12). In each case, we can make the same general observations that were made in the previous example. The region–merge segmentation (Figure 6.13) is quite fragmented in heavy texture and at strong intensity boundaries. The segmentations produced by the localized one–dimensional clustering algorithm (Figures 6.14 and 6.15) are both very overfragmented in foliage and miss important object boundaries. The high sensitivity segmentation (Figure 6.14) completely misses the exterior boundary of the tree, while the very–high sensitivity segmentation (Figure 6.15) produces an approximation to this boundary at the cost of increasing the fragmentation of other textured areas of the image. The GOLDIE IDDS segmentation (Figure 6.16) again shows very good correspondence to scene content, although the occlusion boundary between house roof and garage roof that had been present in the other segmentations is not present here.

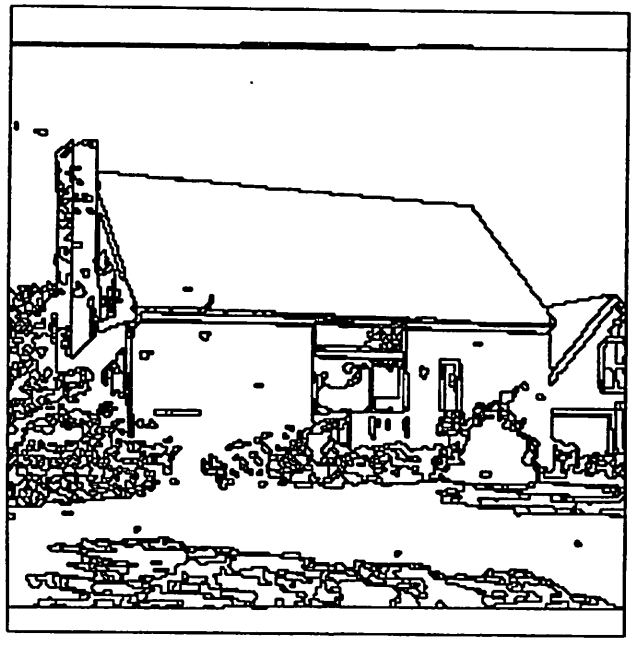


Figure 6.9: Localized One–Dimensional Segmentation of House 15 (Very–High)

The very–low (53 regions), low (154 regions), and medium (231 regions) sensitivities of this algorithm are useless for the segmentation of this particular image, although in other types of images these settings can be quite useful. In the high sensitivity segmentation (Figure 6.8, 485 regions), some of the major scene boundaries are represented on the region boundaries (e.g. house wall, roof), but many others (e.g. chimney, window, and door) are missing or incomplete. In addition, the use of the high sensitivity also introduces a significant amount of fragmentation in the textured areas of the image. If we look at the segmentation produced at very–high sensitivity (Figure 6.9, 842 regions), we see that this data is actually less representative of scene content than was the high sensitivity segmentation. Not only has the fragmentation increased, but the arbitrary placement of the sector boundaries has led to the elimination of a number of regions that should be present in the bushes in front of the house, and thus leads to a sector boundary merge between the wall and grass regions of the segmentation.

This variability of the response of this algorithm to different sensitivity settings is a significant problem. Since there are no automatic evaluation mechanisms built into the algorithm itself, it is often necessary for a user to apply the algorithm at several different sensitivities and then empirically choose the segmentation that is most appropriate for the interpretation processes. For the remaining examples in this section we will do just that; in order to show the best results for comparison, we will show the segmentation produced by the localized one–dimensional clustering algorithm only at the sensitivity for which the segmentation appears to be most appropriate for interpretation.

A significant problem with this localized one–dimensional histogram clustering algorithm is that although the cluster selection is localized to local image areas, the sensitivity settings and boundary merge parameters are applied globally. Thus, even though the cluster selection process is applied with respect to local image *data*, the segmentation process is not truly sensitive to local image *context*. The sector boundaries are predefined, and thus they do not in any way

represent boundaries where the underlying image characteristics change. Therefore, both textured and nontextured areas can be present within a particular sector, and there is no way to make use of any specialized knowledge or control within any of the individual sectors.

In contrast to the segmentations produced by these global segmentation algorithms, Figures 6.10 and 6.11 show the result of the invocation of the GOLDIE IDDS schema over the image of Figure 6.1 (containing 131 regions). This segmentation shows regions that correspond very well to the individual semantic objects in the scene. Because of the intermediate–level evaluation that has been performed by the IDDS schema, the regions of this segmentation have been created and merged with respect to local image context. There are a few obvious errors in the segmentation (e.g. the horizontal line on the house wall that results from the line insertion process), but for the most part, the region boundaries of this segmentation correspond quite well to perceived object boundaries in the scene. Note, however, that this segmentation was produced without any *a priori* constraints, and the nature of this segmentation reflects only the default expectations of the IDDS schema about the interpretation process (i.e. regions should be relatively large and should satisfy the evaluation criteria that are specified as a function of local image context). If the interpretation goal for this image was to identify the particular species of bush, this segmentation would be highly overmerged. With respect to the default condition of identifying all "significant" objects in an outdoor scene, this appears to be a high quality segmentation.

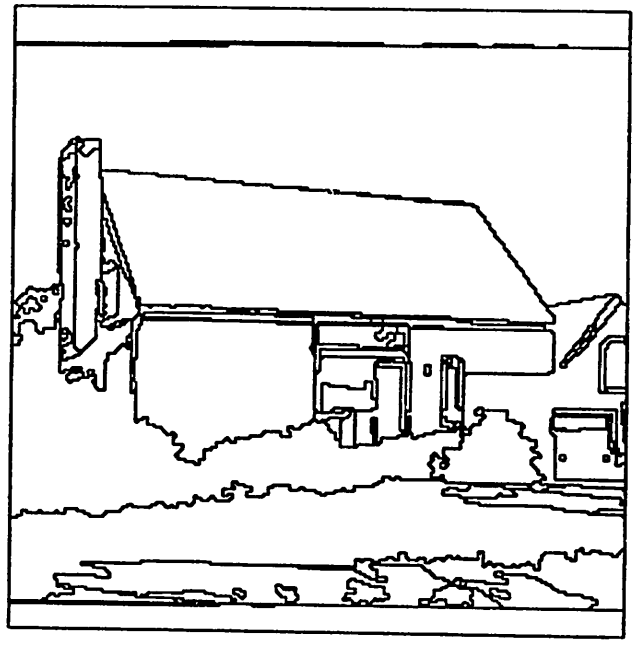


Figure 6.10: GOLDIE IDDS Schema Segmentation of House 15

Figures 6.13 through 6.16 compare these three segmentation processes on a different house image (Figure 6.12). In each case, we can make the same general observations that were made in the previous example. The region–merge segmentation (Figure 6.13) is quite fragmented in heavy texture and at strong intensity boundaries. The segmentations produced by the localized one–dimensional clustering algorithm (Figures 6.14 and 6.15) are both very overfragmented in foliage and miss important object boundaries. The high sensitivity segmentation (Figure 6.14) completely misses the exterior boundary of the tree, while the very–high sensitivity segmentation (Figure 6.15) produces an approximation to this boundary at the cost of increasing the fragmentation of other textured areas of the image. The GOLDIE IDDS segmentation (Figure 6.16) again shows very good correspondence to scene content, although the occlusion boundary between house roof and garage roof that had been present in the other segmentations is not present here.

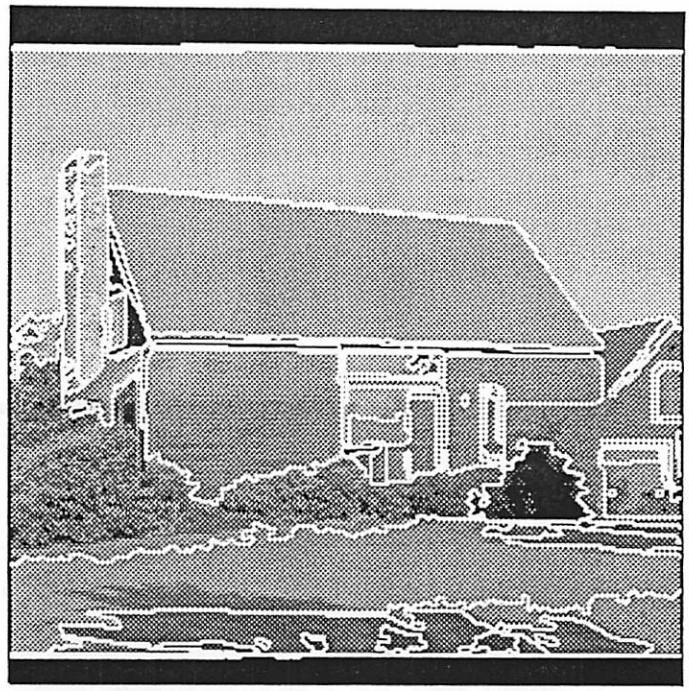


Figure 6.11: House 15 Image with IDDS Segmentation

Although we will not discuss the remaining examples in detail, Figures 6.17 through 6.36 provide the same comparison between the three segmentation algorithms for a variety of images. By providing this extensive set of examples, we want to emphasize the point that the concept of intermediate–level control that has been implemented in GOLDIE does not in any way mean that the system must be expectation–driven. Rather, the data–driven processing exemplified by the IDDS schema may be used without high–level control to produce intermediate–level data that is quite representative of image content.

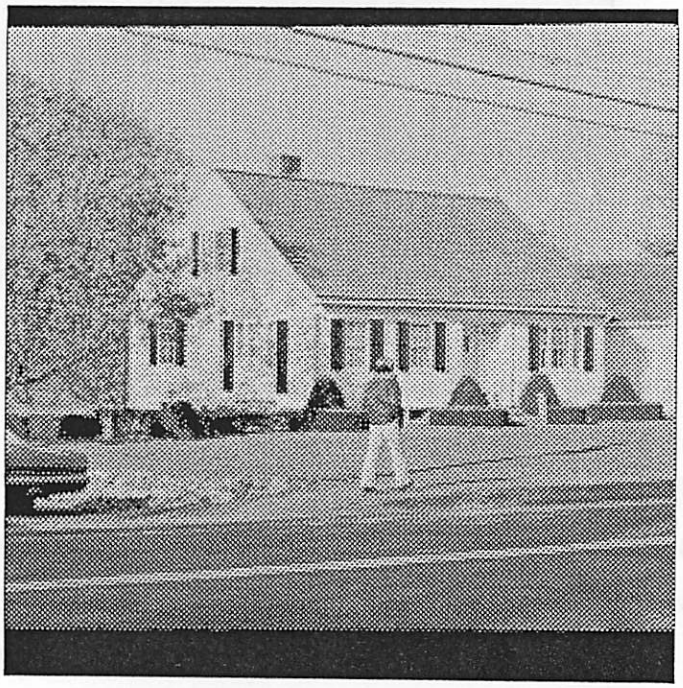


Figure 6.12: Outdoor House Scene (House 1)

Figure 6.13: Region–Merge Segmentation of House 1

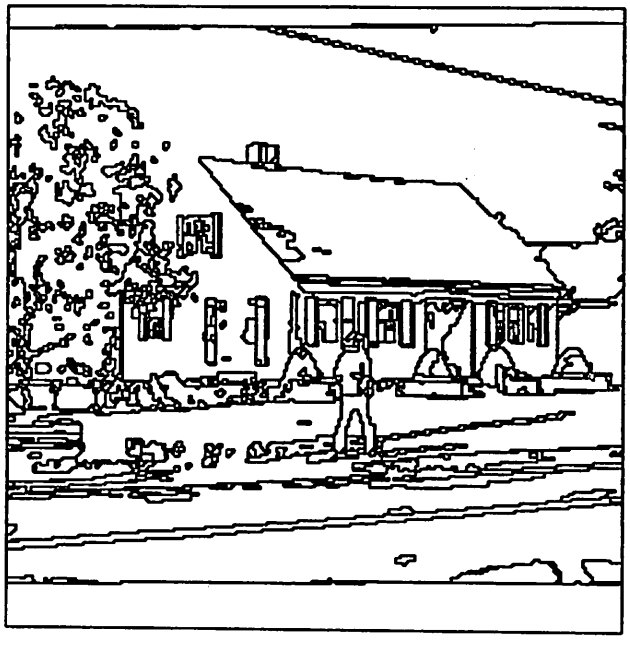


Figure 6.14: Localized One–Dimensional Segmentation of House 1 (High)

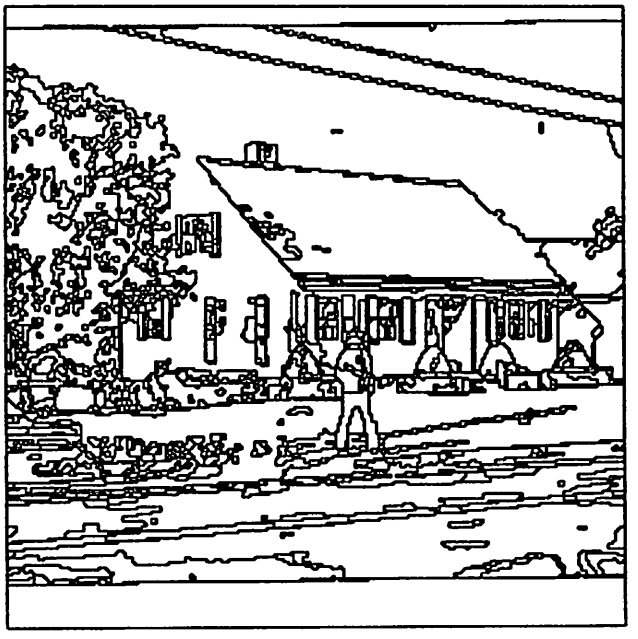


Figure 6.15: Localized One–Dimensional Segmentation of House 1 (Very–High)

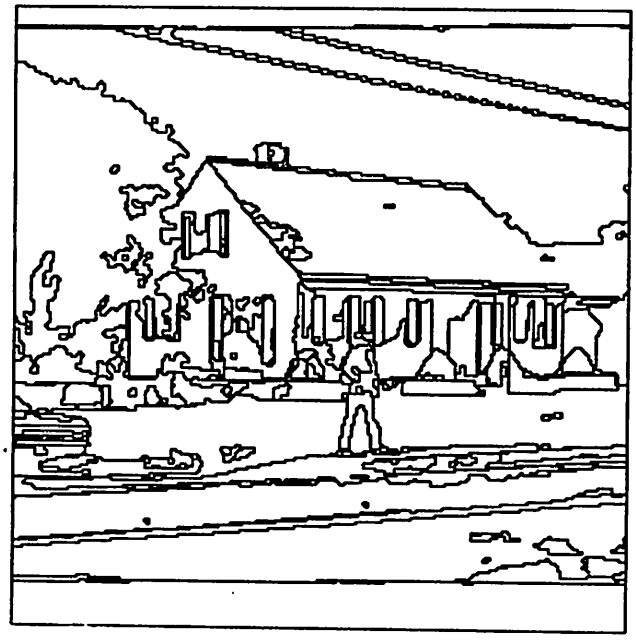


Figure 6.16: GOLDIE IDDS Schema Segmentation of House 1

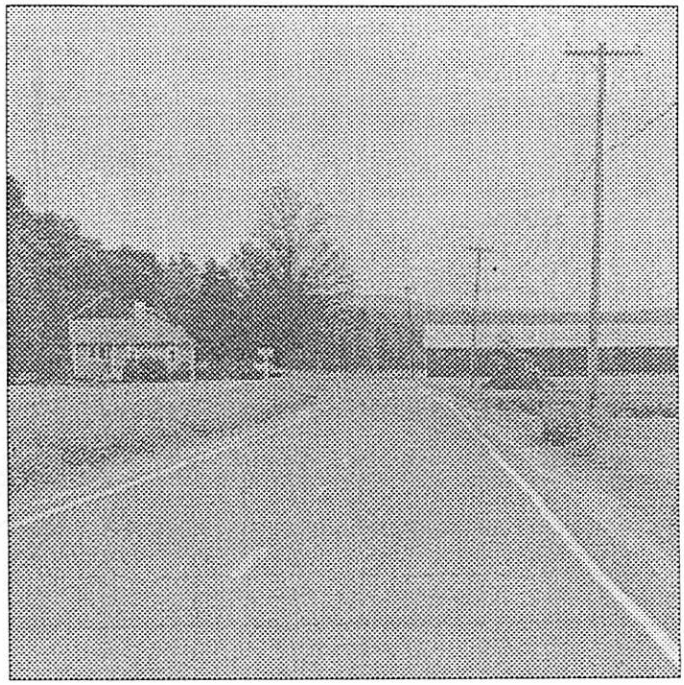


Figure 6.17: Outdoor Road Scene (Road 16)

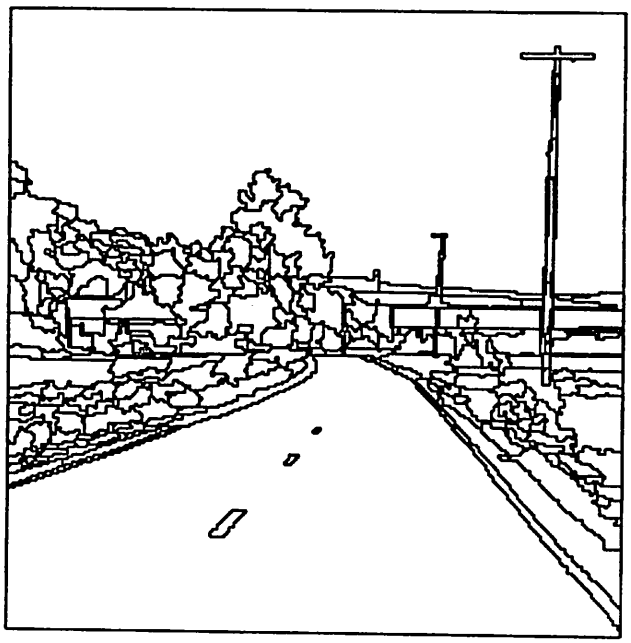


Figure 6.18: Region–Merge Segmentation of Road 16

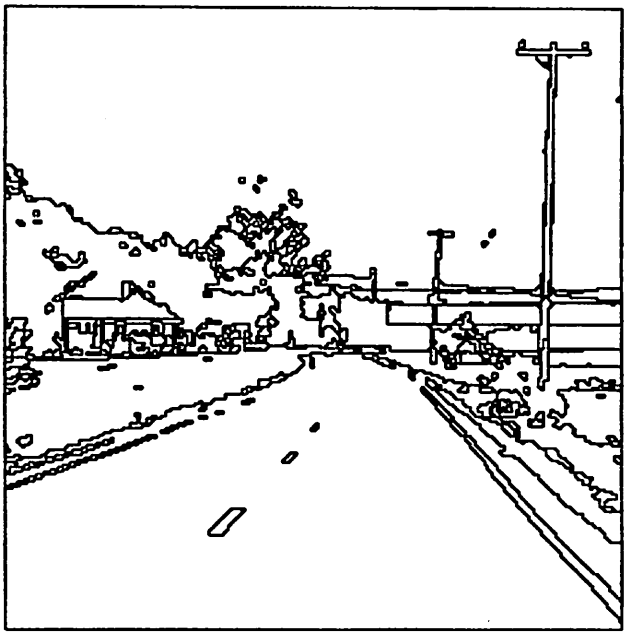


Figure 6.19: Localized One–Dimensional Segmentation of Road 16 (Very–High)

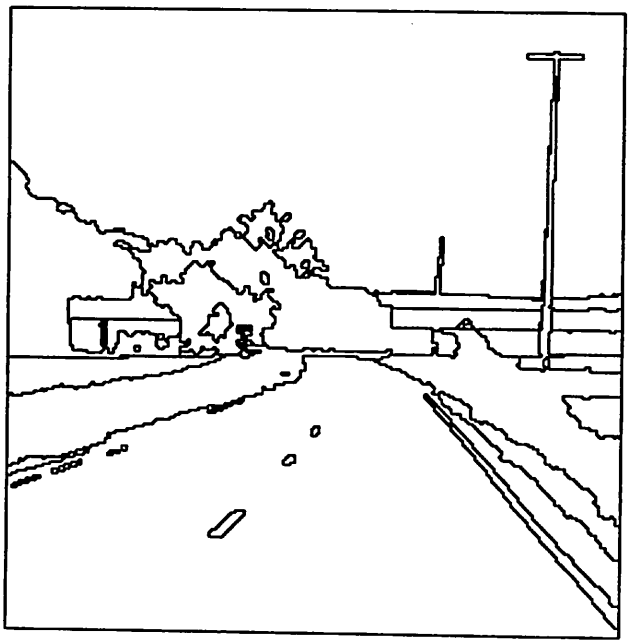


Figure 6.20: GOLDIE IDDS Schema Segmentation of Road 16

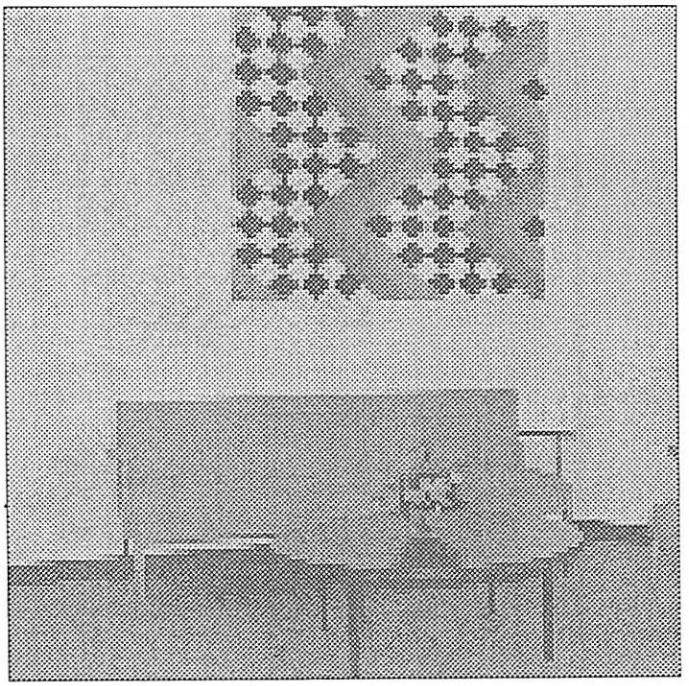


Figure 6.21: Indoor Scene (Room 50)

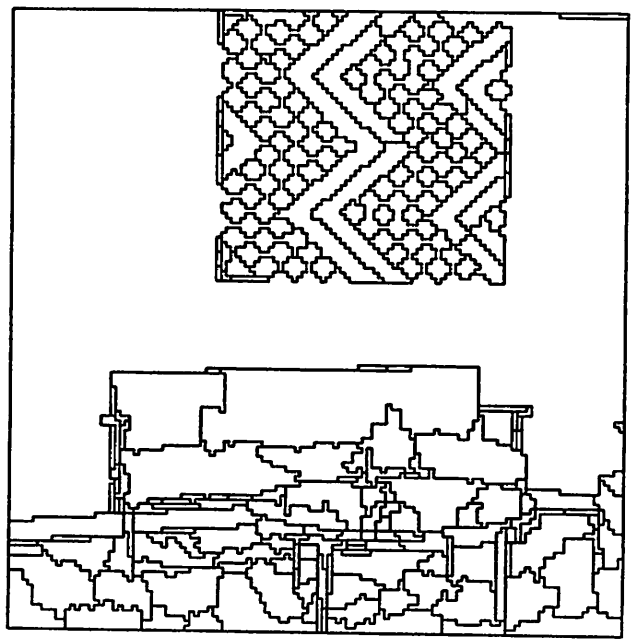


Figure 6.22: Region–Merge Segmentation of Room 50

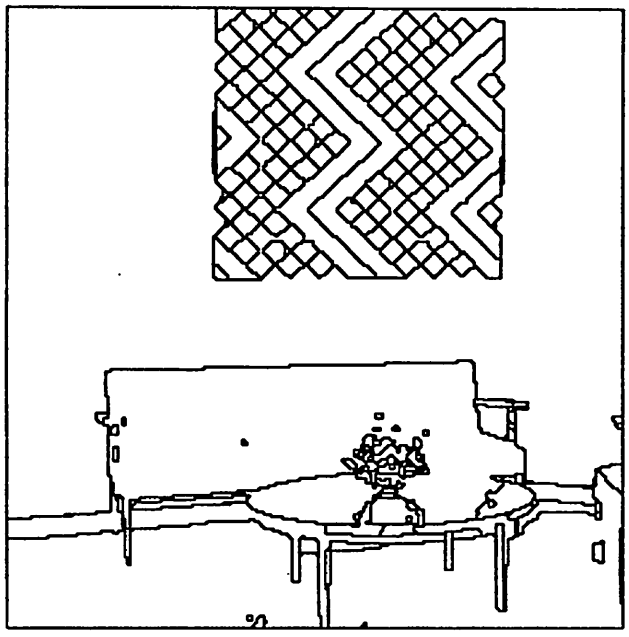


Figure 6.23: Localized One–Dimensional Segmentation of Room 50 (High)

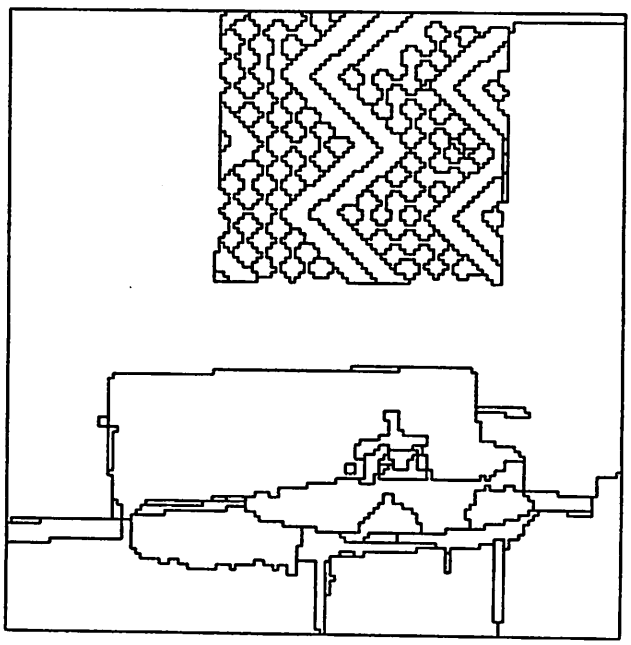


Figure 6.24: GOLDIE IDDS Schema Segmentation of Room 50

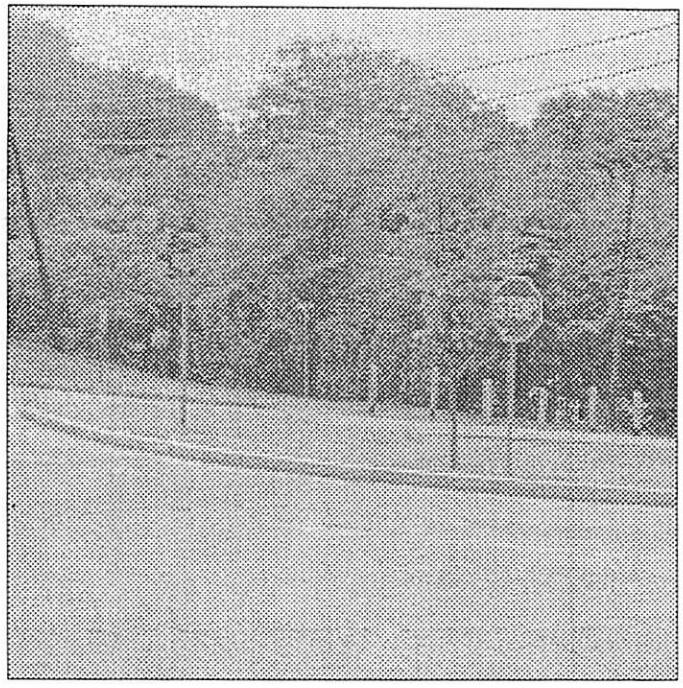


Figure 6.25: Outdoor Road Scene (Road 1)

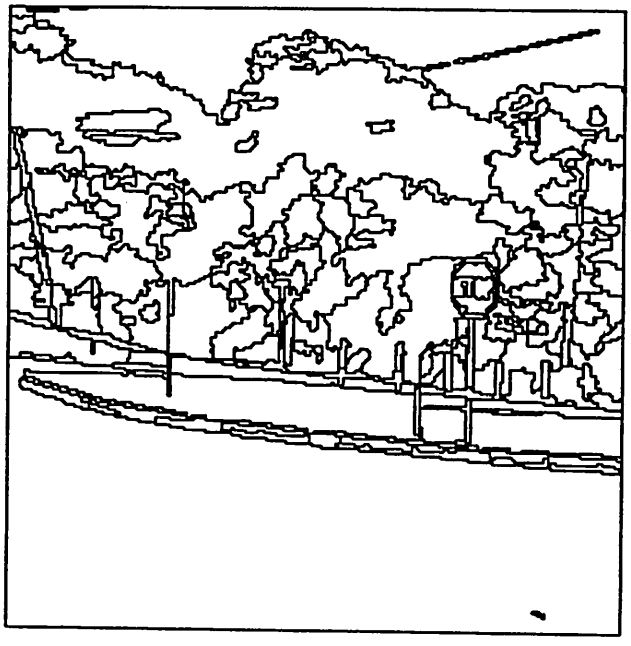


Figure 6.26: Region–Merge Segmentation of Road 1

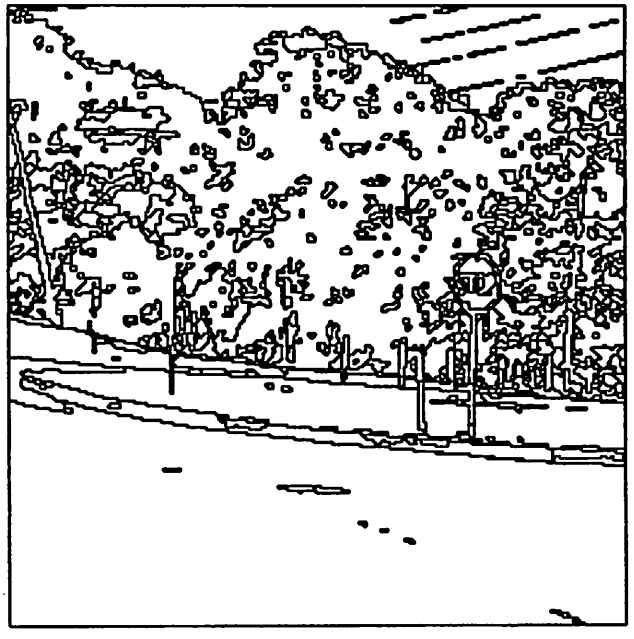


Figure 6.27: Localized One–Dimensional Segmentation of Road 1 (Very–High)

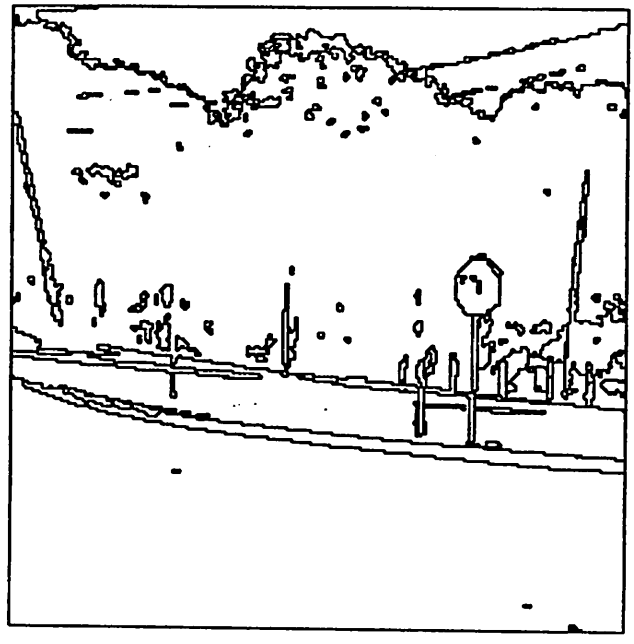


Figure 6.28: GOLDIE IDDS Schema Segmentation of Road 1

Figure 6.29: Outdoor Road Scene (Road 25)

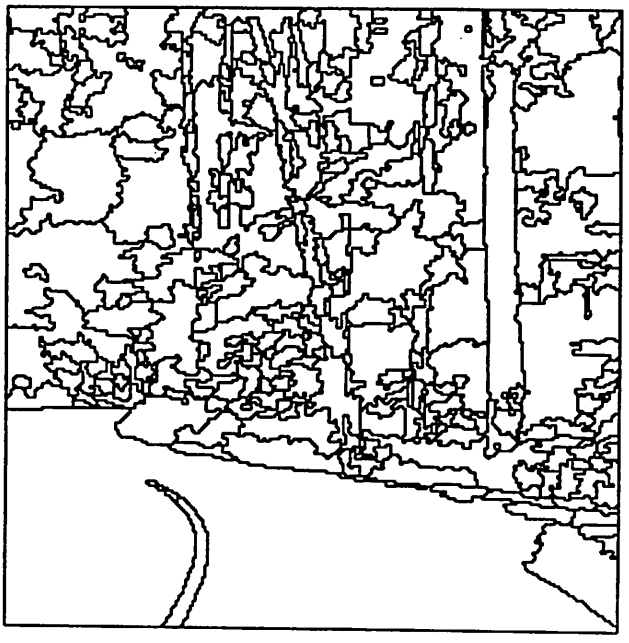


Figure 6.30: Region–Merge Segmentation of Road 25

Figure 6.31: Localized One–Dimensional Segmentation of Road 25 (Very–High)

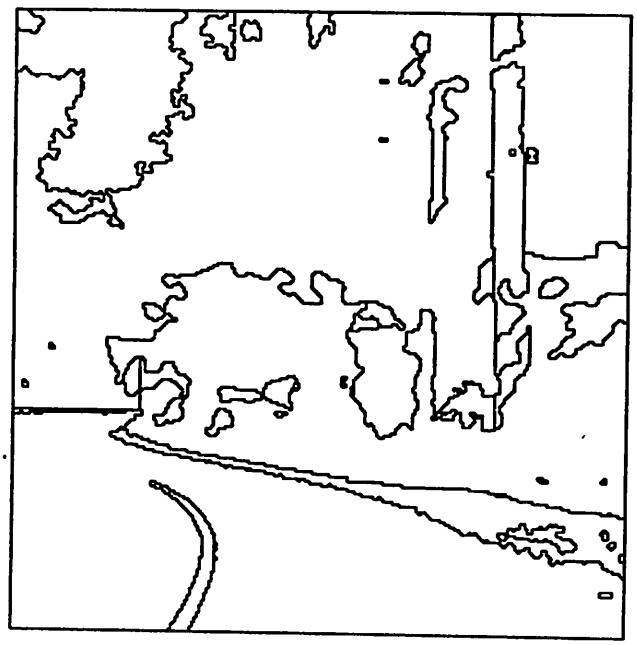


Figure 6.32: GOLDIE IDDS Schema Segmentation of Road 25

Figure 6.33: Outdoor House Scene (House 7)

Figure 6.34: Region–Merge Segmentation of House 7

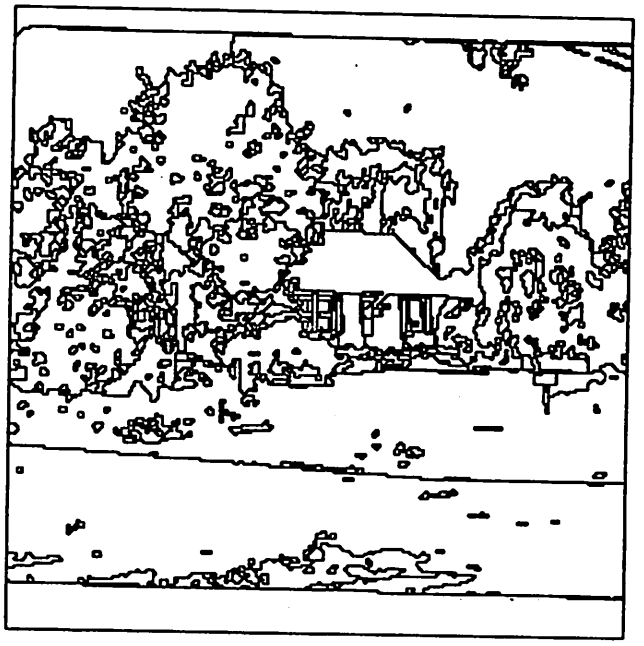


Figure 6.35: Localized One–Dimensional Segmentation of House 7 (High)

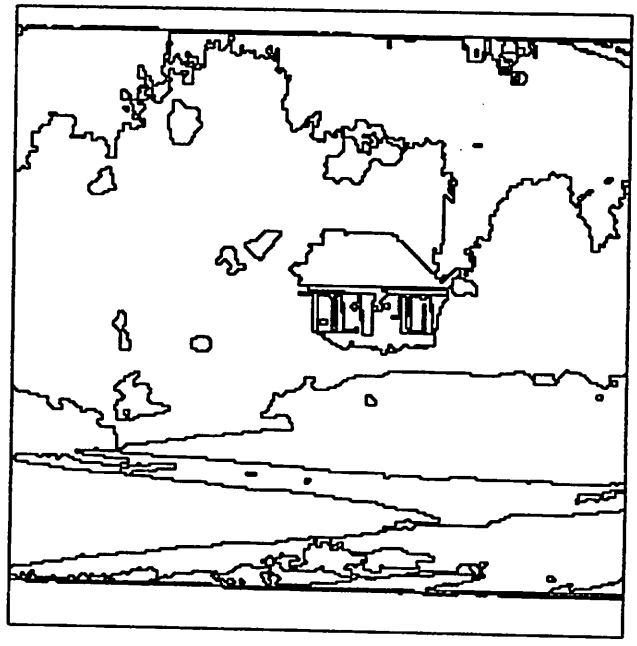


Figure 6.36: GOLDIE IDDS Schema Segmentation of House 7

6.2.2 Intermediate–Level Control in the IDDS Schema

As demonstrated in the previous section, the IDDS schema of GOLDIE can serve as a high quality segmentation system independent of any high level control. A number of factors contribute to the quality of these results:

- local focus and the tailoring of low and intermediate process specifications to the image characteristics of the local neighborhood,
- intermediate–level data–oriented classification and evaluation of tokens,
- intermediate–level knowledge of the appropriate image features, segmentation algorithms, and sensitivity parameters to use in the extraction of tokens,
- integration of information provided by different intermediate–level representations of image events (i.e. regions and lines), and
- a hypothesize–and–test mechanism that allows the schema to explore a variety of alternative processing paths.

In this section, we shall examine the various strategies employed by the IDDS schema during the production of an initial segmentation, and demonstrate the manner in which the schema control mechanism allows this schema to utilize the other schemas of GOLDIE to select the most useful processing path. By tracing the behavior of the schema in the production of the segmentation shown in Figure 6.10, it becomes possible to see not only the actual stages of processing, but also the way in which intermediate–level knowledge is utilized to control the processing. Although the discussion may at times seem overly detailed, it is important understand how the intermediate–level knowledge and control structures of GOLDIE can interact in a complex manner to produce high quality intermediate–level data.

As the unconstrained IDDS schema instance is invoked, it initially forms a contract with an instance of the region–segmentation schema to obtain a "best

Table 6.1: Segmentation Process Specifications from IDDS Schema

Figure	Image Feature	Segmentation Algorithm	Sensitivity Setting	Evaluation Score
6.37	Intensity	Global 1D	0.5	0.62
6.38	Red	Global 1D	0.5	0.66
6.39	Itwob	Global 1D	0.5	0.62
6.40	Intensity/Hue	Global 2D	0.5	0.88
6.41	Slfeat	Global 1D	0.5	0.41
6.42	Blue	Global 1D	0.5	0.89
6.43	Green	Global 1D	0.5	0.79
6.44	Normgreen	Global 1D	0.5	0.43

guess" initial segmentation for the entire image, where "best guess" means that the segmentation satisfies or comes closest to the satisfaction of the default `evaluation-constraint` (*cf.* Section 5.4.1). The region–segmentation instance in turn contracts with instances of the region–feature and region–algorithm schemas to obtain the specifications for the segmentation processes. As the segmentations are created, they are evaluated according to this `evaluation-constraint`. Table 6.1 shows the specifications of the segmentation processes that were performed by this instance of the region–segmentation schema instance, and Figures 6.37 through 6.44 show the actual segmentation results.

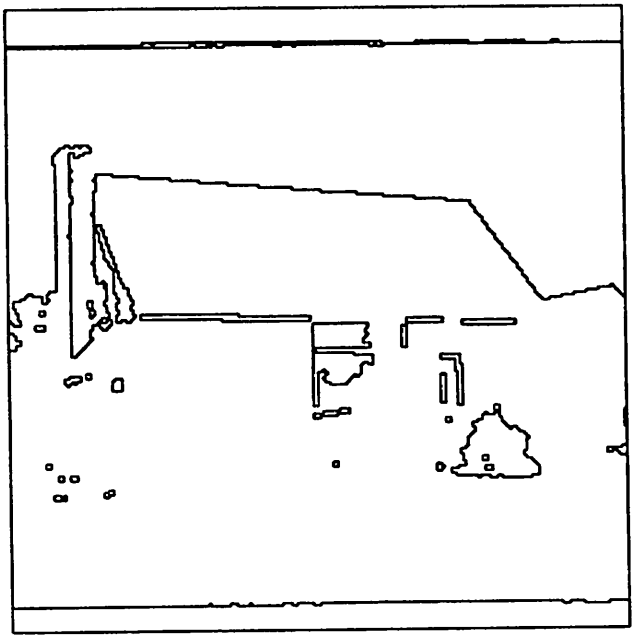


Figure 6.37: Intensity-1D-Plurality Segmentation

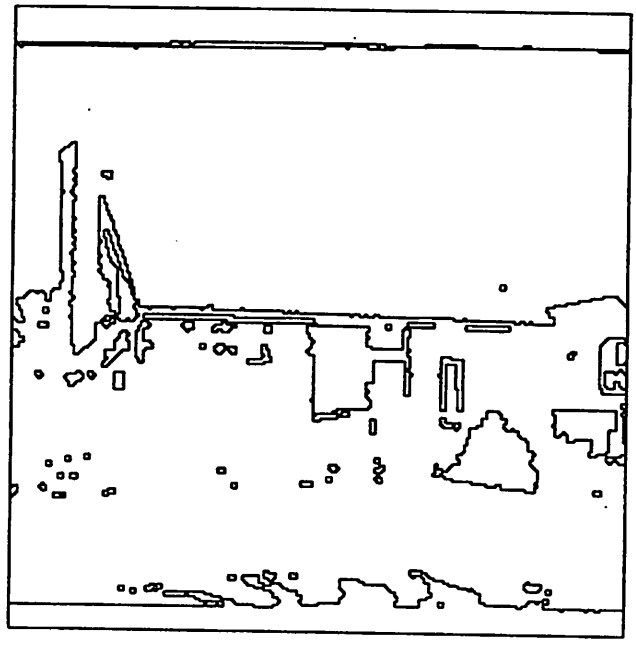


Figure 6.38: Red-1D-Plurality Segmentation

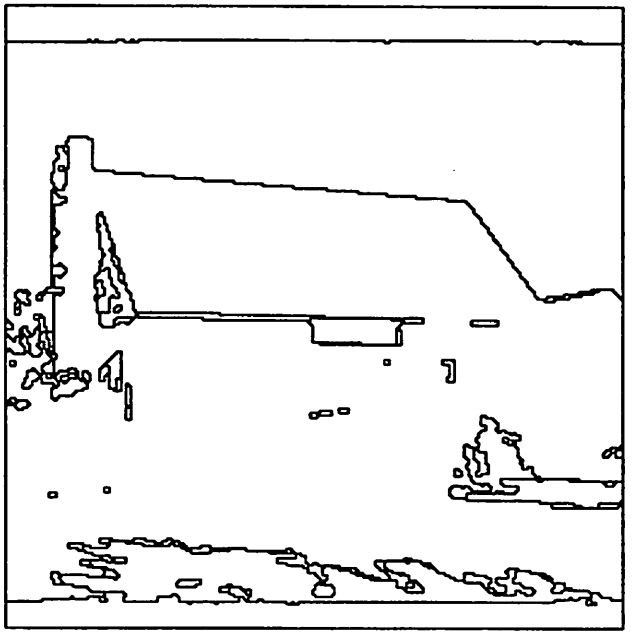


Figure 6.39: Itwob-1D-Plurality Segmentation

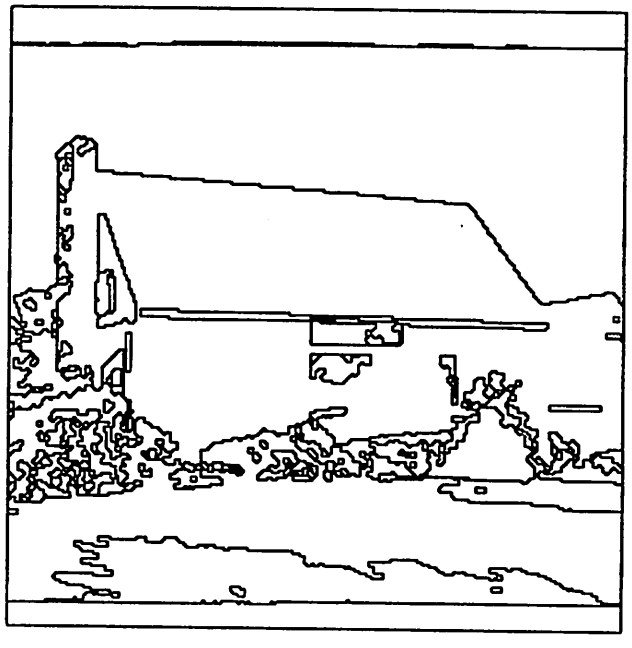


Figure 6.40: Intensity/Hue-2D-Plurality Segmentation

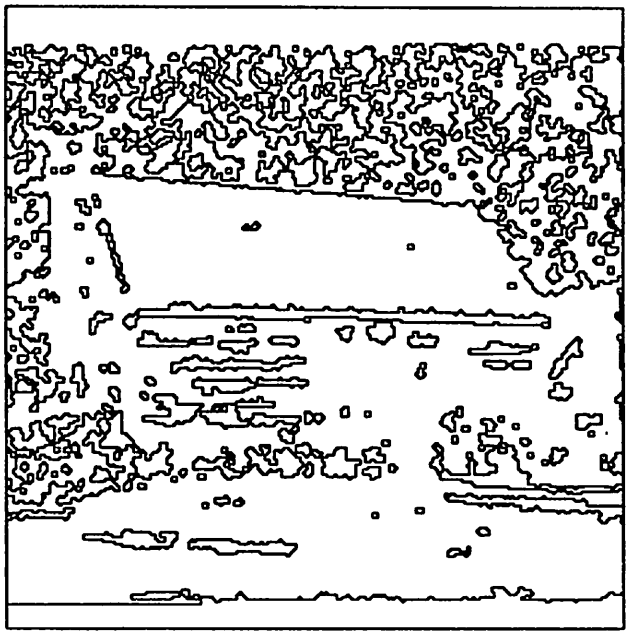


Figure 6.41: Slfeat-1D-Plurality Segmentation

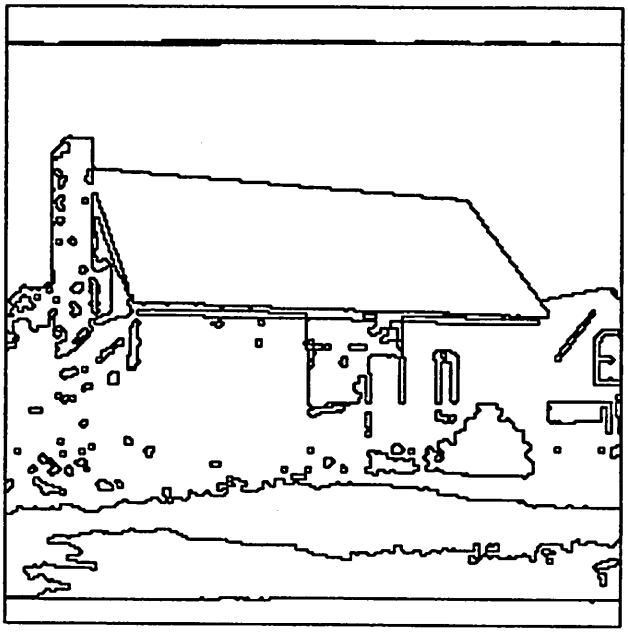


Figure 6.42: Blue-1D-Plurality Segmentation

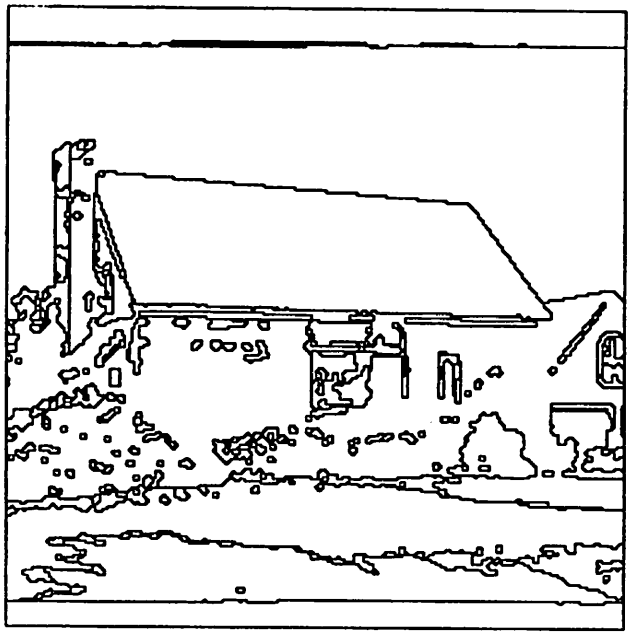


Figure 6.43: Green-1D-Plurality Segmentation

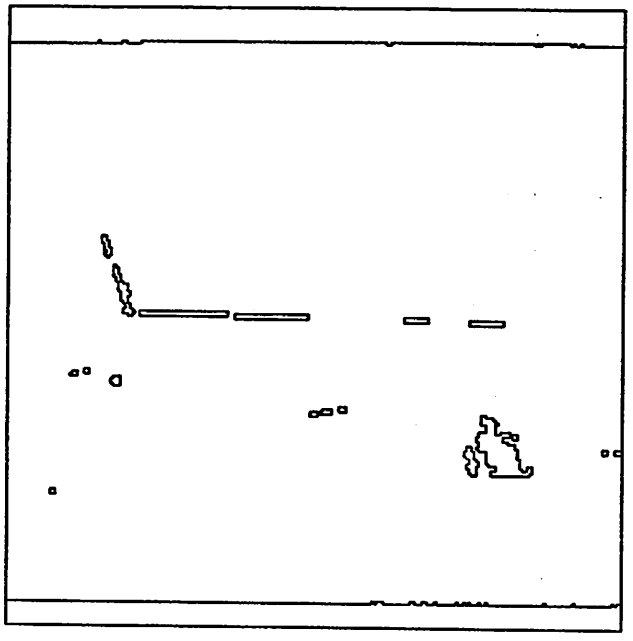


Figure 6.44: Normgreen-1D-Plurality Segmentation

At this early stage of processing, the selection of features, algorithms, and sensitivity settings is necessarily quite general. Therefore, each of the segmentation specifications summarized in Table 6.1 uses the default sensitivity setting, and all but one use the global one–dimensional clustering algorithm. Since none of these segmentations has an evaluation score greater than the acceptance threshold of the default `evaluation-constraint` (0.9), the region–segmentation schema makes use of all the image features proposed by the region–feature schema before returning the segmentation with the best score (Figure 6.42) to the IDDS schema.

Given this initial segmentation the next stage of processing for the IDDS schema instance is to post a goal for line–extraction, thereby producing a set of line tokens for the image (Figure 6.45). Next, the IDDS instance contracts with a line–segmentation instance to insert the long, high–contrast subset of these lines (Figure 6.46) into the region representation, thus producing the set of region tokens in Figure 6.47. Note that this process does not always succeed in the restoration of long lines in the segmentation. The line–insertion process will grow the endpoints of a line only up to a parameterized limit (*cf.* Section 3.8.1). Thus the line representing the forward edge of the chimney in Figure 6.47 can disappear in the connected components process even though the line itself is a long, high contrast line. However, as will be seen, subsequent resegmentation of this area of the image does recover this line.

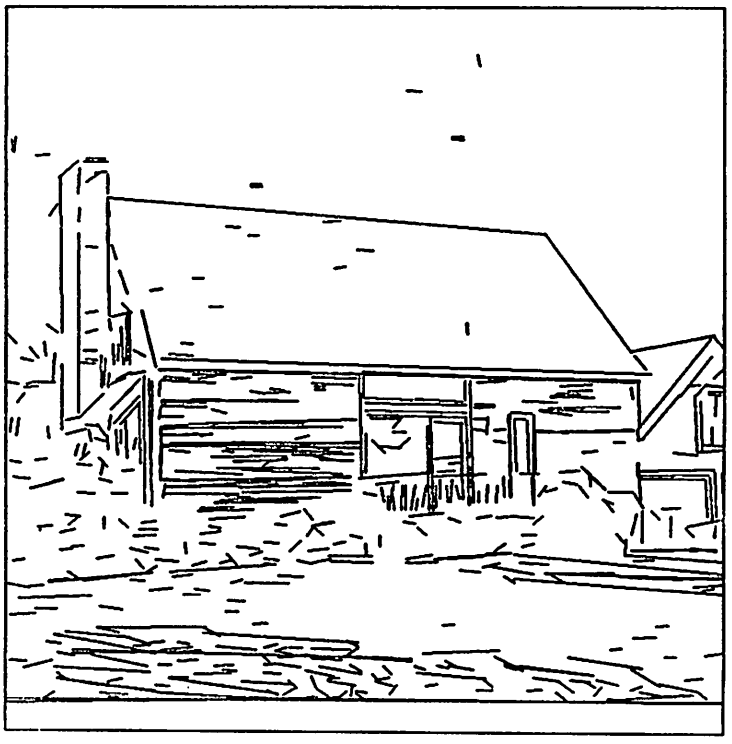


Figure 6.45: Lines for House 15 Image

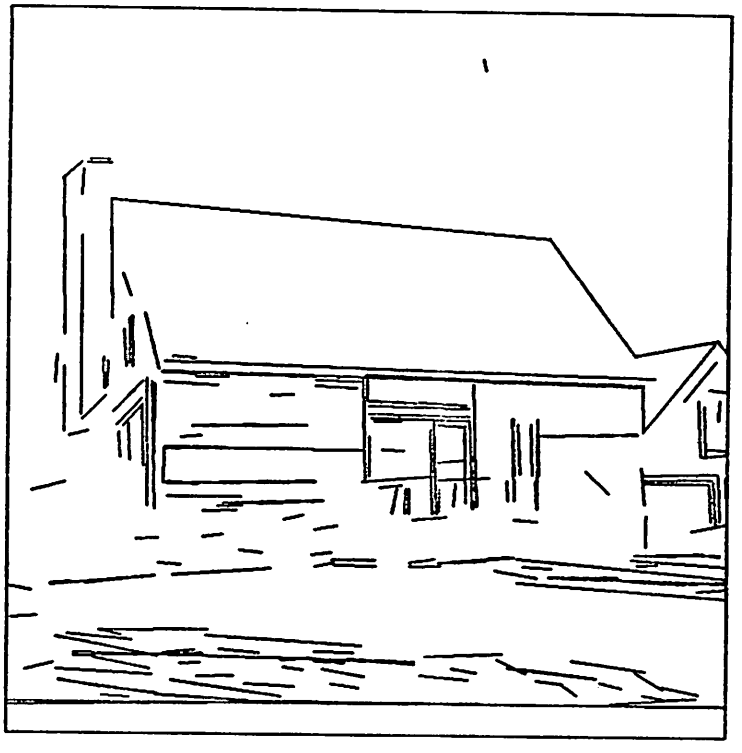


Figure 6.46: Long, High-Contrast Lines for House 15 Image

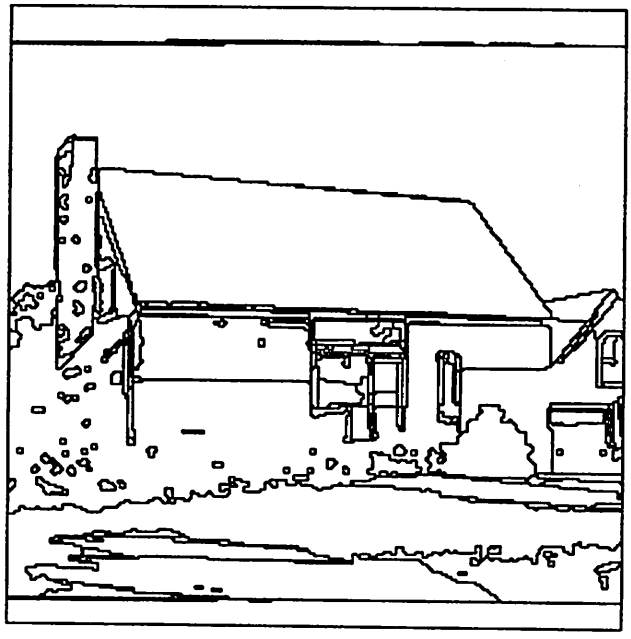


Figure 6.47: Long Line Insertion for House 15 Image

With this initial set of region tokens that have been produced by this early processing, the system now contains a set of intermediate–level entities that can be evaluated by the various intermediate–level knowledge sources. Therefore, the next step for the IDDS instance is to post a goal for region–evaluation in order to produce a set of region–resegmentation hypotheses for this token data. In Figure 6.48 the dark regions are those for which the resegmentation hypothesis exceeds the threshold. Because this set of region tokens represents the entire image, and because there is no *a priori* `region-characteristic` constraint, this particular instantiation of the region–evaluation schema uses the default rule set (smooth) to perform the evaluation. An additional goal for intermediate–level semantic evaluation is also posted, producing a set of semantic hypotheses for this set of region tokens so that any other schemas that will be operating on this data can make use of potential semantic information about the characteristics of the regions.

Given a localized image context, it now becomes possible to recursively invoke the IDDS schema over the set of resegmentation regions to refine the segmentation data with respect to the local image characteristics. As an example, the recursive invocation of the IDDS schema over the region corresponding to the shadowed grass in the bottom of Figure 6.1 is made according to the goal specification in Figure 6.49. Here, the intermediate–level evaluation has determined both that there is evidence that resegmentation of this region is required and that there is also evidence for the existence of grass and/or road in this part of the image. Because grass is a textured object and road is a non–textured object, no `region-characteristic` constraint can be specified.

Using this constraint information, the newly invoked instance of the IDDS schema invokes a new instance of the region–segmentation schema which then in turn activates an instance of the region–feature schema. The region–feature instance uses the `object-label` constraint to initially propose the segmentation features Objfeat–bush–grass, Objfeat–bush–tree, Green, and Hue (Road is not

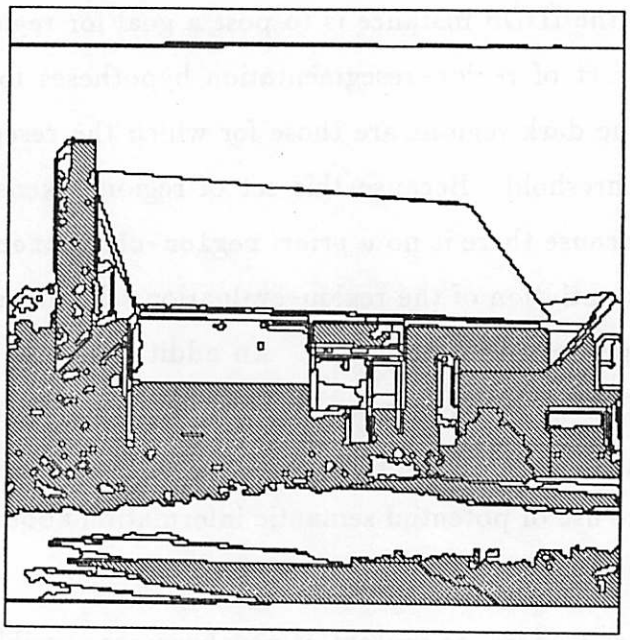


Figure 6.48: Resegmentation Hypothesis Regions

Region-Segmentation
'((*region–token* . Region-01264)
(*object–labels* . (grass road))

Figure 6.49: Goal for the Resegmentation of Grass Region

an ILTM object for which reliable statistics exist, so there is no Objfeat–grass–road feature defined). After evaluating the histograms for these four features, the region–feature schema ranks the utility of the features and returns the ordered list '(Hue Objfeat–bush–grass Green) to the region–segmentation schema. Since the Objfeat–bush–tree feature displayed only one peak with this parameter setting, it was inappropriate for segmentation and eliminated from the list.

The region–segmentation schema then invokes the region–algorithm schema to select the most appropriate algorithm for segmentation. Because the specified image-feature is Hue, the region is relatively non–textured, and the objects in question are represented in ILTM as being relatively unstructured, this schema instance proposes the global one–dimensional clustering algorithm with a sensitivity setting of 0.3. However, in attempting to perform a segmentation with these specifications, the region–segmentation schema determines that only one cluster is identified in the feature histogram. Therefore the region–algorithm is queried for another algorithm. This time the region–algorithm schema responds with global one–dimensional clustering at a sensitivity setting of 0.5, and the resulting segmentation is shown in Figure 6.50. Even though the overall change in the region definition is small (several dark areas on the right side of the region have been isolated), this new segmentation produces an evaluation score of 1.0. This evaluation score exceeds the acceptability threshold of the default `evaluation-constraint`, and the segmentation is immediately accepted by the region–segmentation schema. Region evaluation is then performed, and since no further region segmentation is required, region merging is performed and the results are returned to the parent instance of the IDDS schema.

In this example, the GOLDIE schemas have made use of rather minimal constraints to produce a resegmentation of a region token; in other situations the IDDS schema is capable of expressing a more complete set of constraints on the goal for resegmentation. For the region in the middle of the image that represents bush and house wall, the IDDS schema posts the goal specified in Figure 6.51.

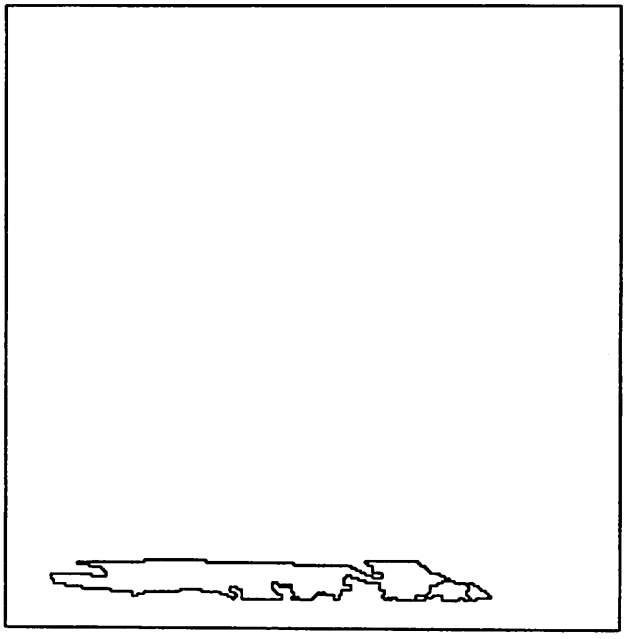


Figure 6.50: Resegmentation of Grass Region

Region-Segmentation
'((*region–token* . Region-01120)
(*region–characteristic* . texture)
(*region-resolution* . low)
(*object–labels* . (bush tree))

Figure 6.51: Goal for the Resegmentation of the Bush/Wall Region

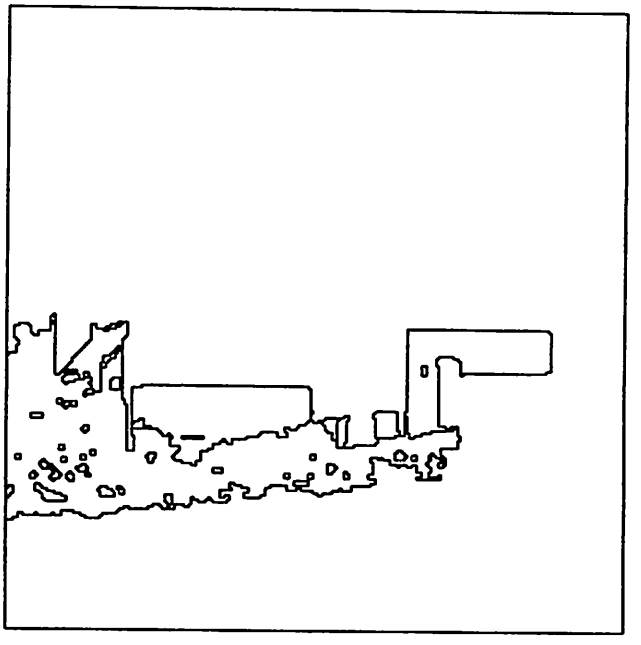


Figure 6.52: Resegmentation of Bush/Wall Region

Because the region is relatively large and highly textured, a high resolution segmentation of this region would be expected to produce a great deal of fragmentation. Therefore this goal is posted with the `region-resolution` constraint "low". The specification of this constraint is designed to indicate to the responding schema instance to use low sensitivity settings (if possible) in the resegmentation process. In this way, GOLDIE can subdivide this region into a small number of new regions that will potentially provide greater information about local context. As shown in Figure 6.52, the recursive invocation of the IDDS schema does indeed partition this area into regions representing bush and wall through the use of a segmentation process specified by the Hue image feature, global one-dimensional clustering, and sensitivity setting of 0.3. Since this resegmentation has an acceptable evaluation score (0.99), the IDDS schema then evaluates each of these new regions for resegmentation. In this case, the large region corresponding to bush still requires resegmentation, so an additional goal is posted for yet

Region-Segmentation
'((*region–token* . Region-01395)
(*region–characteristic* . texture)
(*region-resolution* . low)
(*object–labels* . (bush))

Figure 6.53: Goal for the Resegmentation of the Bush Region

another recursive invocation of the schema (Figure 6.53).

In response to this goal, the newest invocation of the IDDS schema attempts to segment this region, and after a number of attempts, produces the segmentation of Figure 6.54 using Green, thresholding, and sensitivity of 0.3. In this case, however, the segmentation that is produced is almost identical to the original region. The fact that a region evaluates for resegmentation does not necessarily mean that it is possible to produce an acceptable resegmentation. If the best possible resegmentation is almost identical (i.e. 95% overlap with the original region), the IDDS schema concludes that no further resegmentation would be profitable under the current goal constraints, and terminates recursive invocation regardless of the evaluation scores for the newly created regions.

This same form of recursive resegmentation is performed for the other regions of the initial segmentation that have strong resegmentation hypotheses, and once all of these recursive iterations of the IDDS schema have completed, the set of region tokens in ISTM is represented by the segmentation shown in Figure 6.55. This data may then be processed to perform the final top–level region merging. First the regions are processed by the region–merge schema using the default constraints, producing the segmentation of Figure 6.56. The IDDS schema instance then invokes the region–merge schema again, this time specifying the `texture-merge` constraint so that the merge threshold is lowered for small regions that are adjacent to large textured regions (Figure 6.57). A final invocation of the region–merge schema specifying the `boundary-merge` constraint is

used to lower the merge threshold for long, thin regions and produces the final result of the IDDS schema shown in Figure 6.58[2].

It is the intermediate–level control of data–directed processing that makes the output of this schema so different from that of the other algorithms that were discussed in the preceding section. Through the specification of goals that are sensitive to local image characteristics, the IDDS schema is able to refine and tune the data to produce intermediate–level tokens that correspond well to the local image characteristics, and can thus serve as appropriate data for interpretation.

6.2.3 Initial Segmentations Produced Under *a priori* Constraints

Although the experiments in the preceding sections have shown the way in which intermediate–level control can be used to control the application of various low and intermediate–level processes, we have not as yet demonstrated the way in which intermediate–level schemas may be constrained to produce data that is

[2]In this particular case, the effects of the boundary merge step were not dramatic; four regions on the front edge of the roof were actually merged.

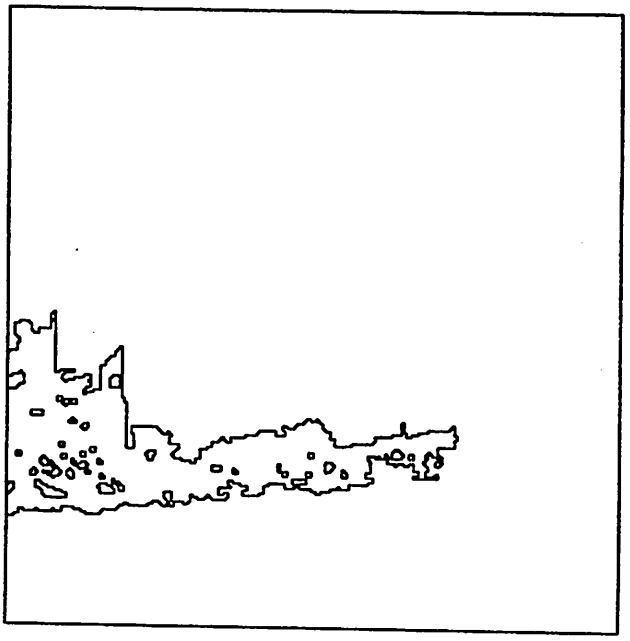


Figure 6.54: Resegmentation of Bush Region

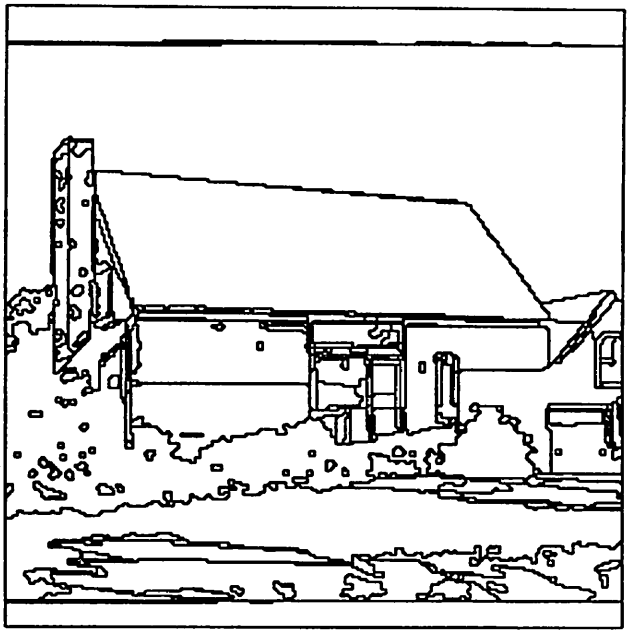


Figure 6.55: Segmentation Prior to Final Merging

appropriate to specific requirements of a higher level process. An example of this form of control is the use of goal constraints to direct the initial segmentation process of the IDDS schema. Through the modification of intermediate–level evaluation criteria, the behavior of a schema instance may be constrained to identify tokens as being either clearly appropriate or clearly inappropriate to the interpretation process. Little or no effort need be expended in the refinement of tokens that are clearly inappropriate, and thus the processing resources of the system may be directed towards those portions of the image that are most valuable to the interpretation process.

An example of the use of the *a priori* constraints is shown in Figures 6.59–6.61 where a house image has been processed by the IDDS schema using the `region-characteristic` constraints smooth, textured and gradient respectively. Using the smooth constraint, all regions are expected to be nearly uniform with respect to the color image features. Thus the segmentation (Figure 6.59) is quite

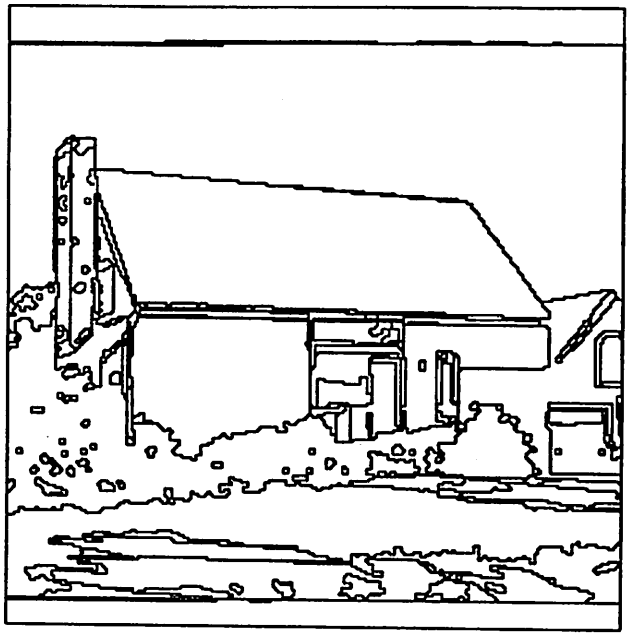


Figure 6.56: Results of Region–Merge Schema with Default Constraint

fragmented. With the texture constraint, regions are expected to be uniform with respect to textural characteristics. In the segmentation (Figure 6.60), we see that the individual textured objects (e.g. tree and bushes) are represented reasonably well, although nearly all of the detailed structure in the house wall has been lost. The gradient segmentation (Figure 6.61) selects regions that globally represent areas of the image that are smoothly varying. Thus, the regions of this segmentation only roughly correspond to underlying scene content, and much of the fine structure has been lost.

This type of *a priori* constraint specification can be used to tune the behavior of the system to certain highly specific purposes. Using as an example the situation presented in the introduction to this chapter, we show how the behavior of the IDDS schema may be constrained to produce initial segmentations for systems with the high–level interpretation goals to produce identification of smooth road areas. Assuming that this interpretation process is running on an autonomous

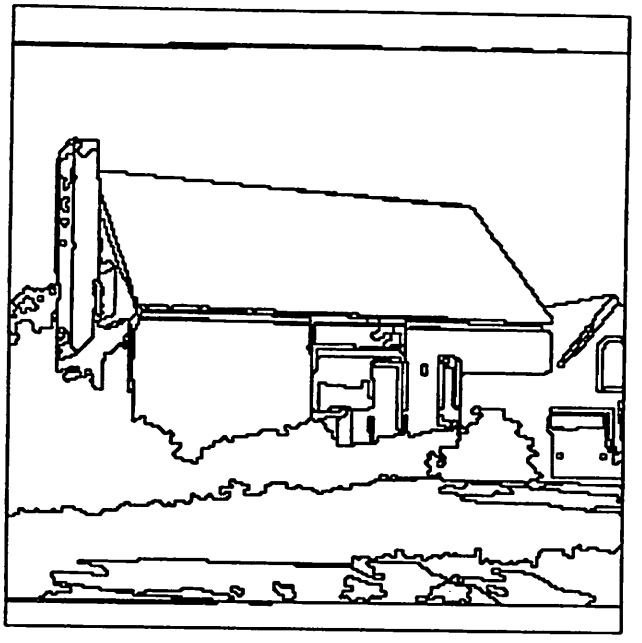


Figure 6.57: Results of Region–Merge Schema with Texture Constraint

vehicle, it is reasonable to assume first that it is not necessary to perform a complete interpretation of the scene, and second that the computation must be kept to a minimum. We could therefore post a goal for the IDDS schema (Figure 6.62) that significantly constrains the segmentation processing.

The constraints in this goal express the high–level need for a segmentation that is coarse, but which roughly identifies the major smooth areas of the image. The `region-token` constraint Region-00002 (the entire image) identifies the area to be segmented, the `segmentation-characteristic` constraint state that the smooth region evaluation rules should be used, and the `segmentation-resolution` constraint indicates and that the segmentation sensitivity settings should be low. Since there is no need to precisely identify regions, the `segmentation-depth` constraint specifies that only one level of recursive invocation of the IDDS schema occur; and since we are interested only in large smooth regions, the `ignore-texture` constraint states that large textured regions should not be considered either for

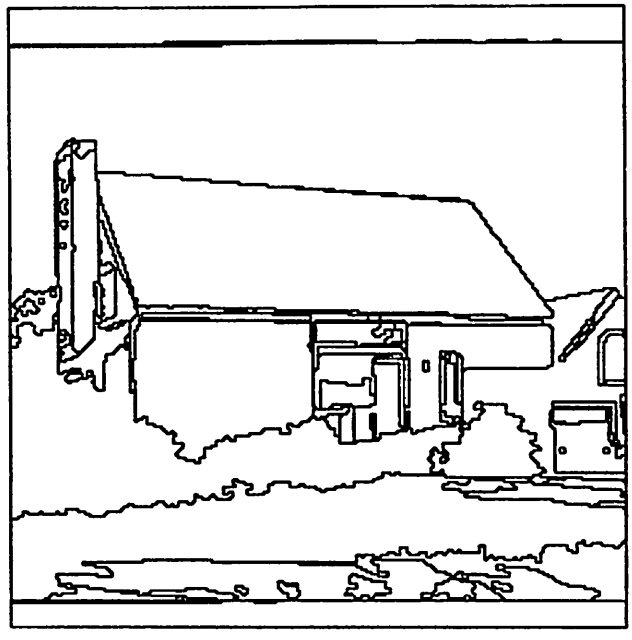


Figure 6.58: Results of IDDS Schema

resegmentation or in the calculation of the evaluation score. Because the interpretation system will only be considering the large regions anyway, there is no need to waste computation by invoking the region–merge schema; thus the `perform-merging` constraint is *nil*. Finally, the `object-label` constraint specifies that the object of interest is road so that any information about road in ILTM may be used in the specification of the segmentation processes.

Using the road images from Section 6.2.1, Figures 6.63, 6.64, and 6.65 show the results of the specification of this goal for the three road images. Here the results of data–directed processing are quite different from the unconstrained segmentations presented earlier. Although these segmentations are coarse, the roads themselves have been clearly discriminated by large regions in the segmentations. There should be no problem for a "quick and dirty" road schema to quickly identify the roads in these images. Note, however, that the specification of this type of goal does not reduce the overall system to a "road finder". What we have

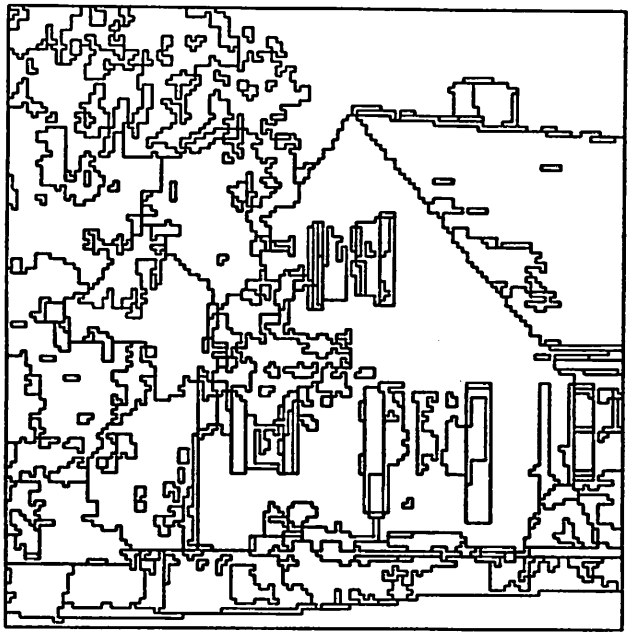


Figure 6.59: IDDS Segmentation With Smooth Constraint

done is to define a particular set of intermediate–level constraints that enable GOLDIE to respond in an efficient manner to a particular type of interpretation need. If, at any time, the interpretation processes require higher quality data (i.e. intermediate–level data that is provides a more precise description of image content), additional goals can be used to obtain the necessary tokens.

6.3 Expectation–Driven Processing

Once an initial segmentation has been produced and partially analyzed by the interpretation system, the high–level requirements for the intermediate level tokens may become much more specific. For example, a semantic labeling process may produce evidence that two or more semantic labels may be assigned to a particular region token. In this case, it becomes reasonable to post a goal to GOLDIE for a resegmentation of the region under the constraint that the token

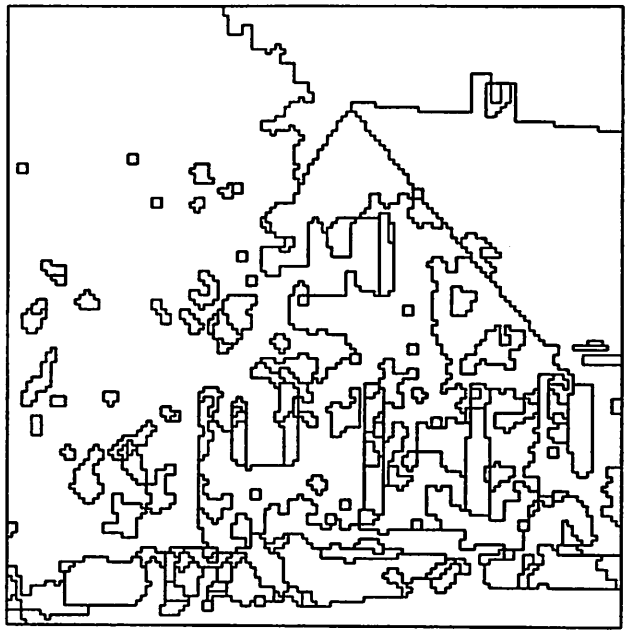


Figure 6.60: IDDS Segmentation With Texture Constraint

be subdivided into areas each representing only one of the specified objects. A more complex situation occurs when there is semantic evidence that a particular object should be present at a particular location in the image, but there is no (set of) token(s) corresponding to that object. In this section we will discuss the ways in which expectation–driven processing can be used to resolve these types of interpretation failures. In both cases, the interpretation system has produced information that may be used by GOLDIE to direct a refinement of the intermediate level data. In this section, we will present a number of examples in which a high–level process could potentially detect a problem in the intermediate–level data and post a constrained goal to GOLDIE for the refinement or resegmentation of a particular subset of this data. We then demonstrate the mechanisms by which the schemas of GOLDIE can utilize the intermediate–level knowledge structures to interpret these high–level goals in terms of intermediate and low–level processes that have the potential to resolve the problem.

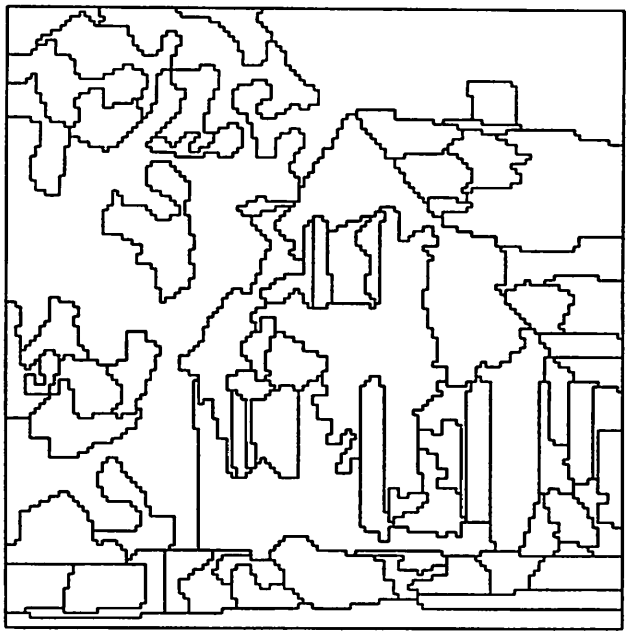


Figure 6.61: IDDS Segmentation With Gradient Constraint

IDDS

'((*region–token* . Region-00002)
(*segmentation–characteristic* . smooth)
(*segmentation–resolution* . low)
(*segmentation–depth* . 2)
(*ignore–texture* . t)
(*perform–merging* . nil)
(*object–labels* . (road))

Figure 6.62: Goal Specification for Road Segmentations

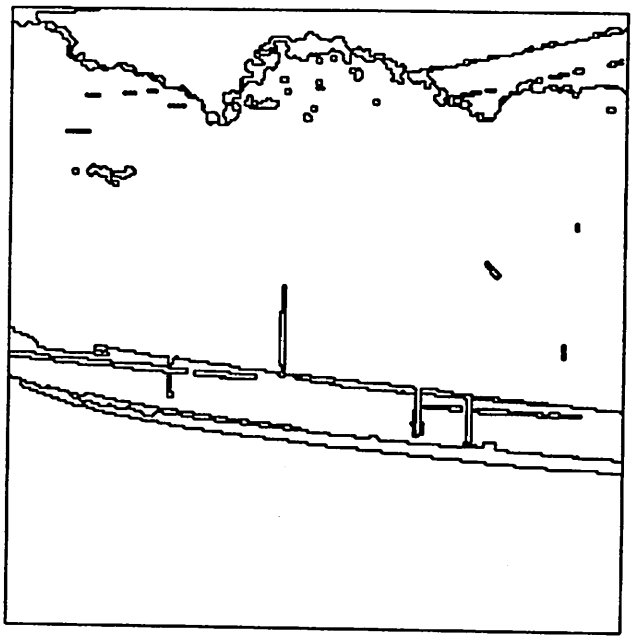


Figure 6.63: Constrained IDDS Segmentation for Road 1

It should be noted, however, that we are dealing with a hypothesized interpretation system. GOLDIE has not as yet been fully integrated with the high–level interpretation schemas in the VISIONS system. Thus, although these examples show the actual results produced by the intermediate–level schemas, the goal specifications themselves are based on a general understanding of the type of problems that can arise in interpretation processing.

6.3.1 Expectation–Driven Processing for Object Extraction

The first example of expectation–driven processing is shown in Figure 6.66 (a subsection of the image in Figure 6.17) where a barn is visible in the foliage behind the house. For some reason, this structure has not been discriminated in the unconstrained data–directed segmentation for this image. If we assume

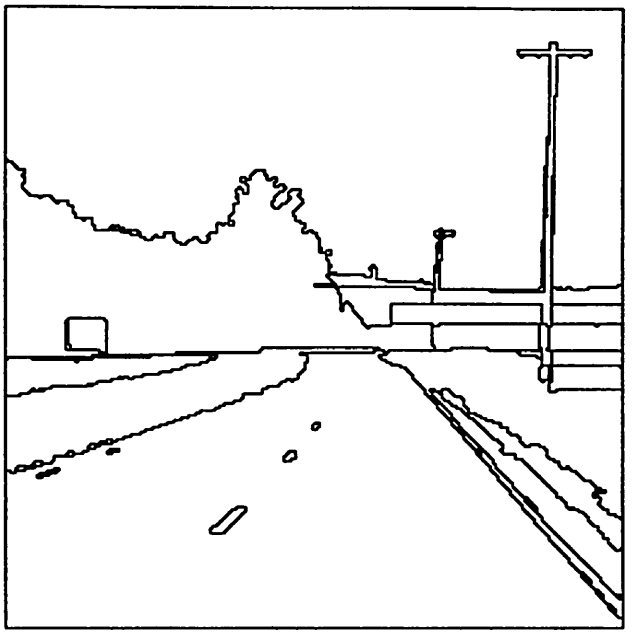


Figure 6.64: Constrained IDDS Segmentation for Road 16

a situation in which a high–level interpretation process had hypothesized the existence of this type of structure in this approximate location, it then becomes possible for the interpretation process to post a goal for the resegmentation of the foliage region to identify any smooth regions in this area. Figure 6.67 shows how this processing has produced a new set of region tokens that clearly discriminate the area of the image that contains the barn. In this example, the schemas of GOLDIE are directed solely by the intermediate–level characterization of *smooth* that was expressed on the resegmentation goal. Since the system currently has no particular knowledge of "barn", it becomes the responsibility of the interpretation process to express the region–segmentation goal according to the set of intermediate–level characterizations that are known to the schemas.

If, however, the semantic objects in question are already represented in ILTM, the interpretation process may express the goal constraints directly. Figure 6.68 shows a particular region of the segmentation from Figure 5.13 in Chapter 5. In

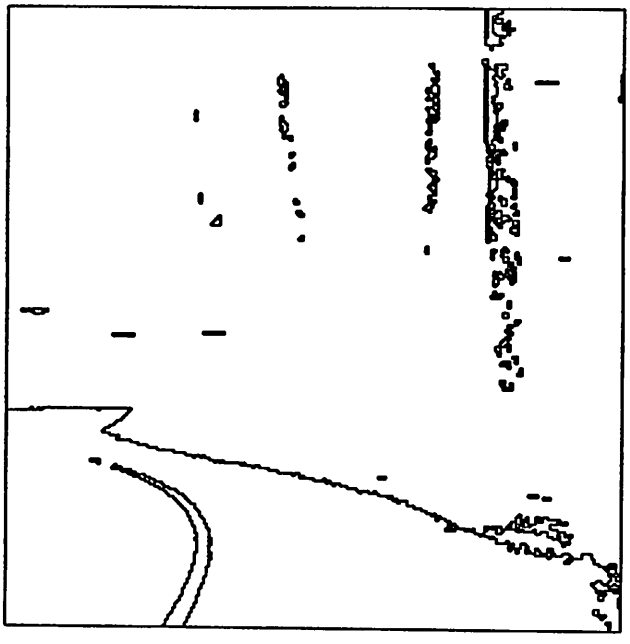


Figure 6.65: Constrained IDDS Segmentation for Road 25

the scene, this is an area of sparse tree foliage in which both sky and foliage are visible. In such a situation, it is often impossible for an interpretation process to assign a unique semantic label to the region. If the interpretation process were to post a goal for the resegmentation of this region, this time with the constraint that `object-labels` was '(tree sky), we would want the region–segmentation schema to use those features and algorithms that would best discriminate regions corresponding to these two types of object. The resulting segmentation would be expected to contain a new set of regions that could be unambiguously labeled. Figure 6.69 shows the the region to be resegmented, and Figure 6.70 shows the set of tokens produced by GOLDIE in response to this goal. In this figure, it can be seen that the segmentation process has split this area into tokens that either correspond to areas that contain mostly sky or correspond primarily to foliage.

Another example of this type of expectation–driven processing is the resegmentation of a bush/grass region shown in Figures 6.71 and6.72. In this case,

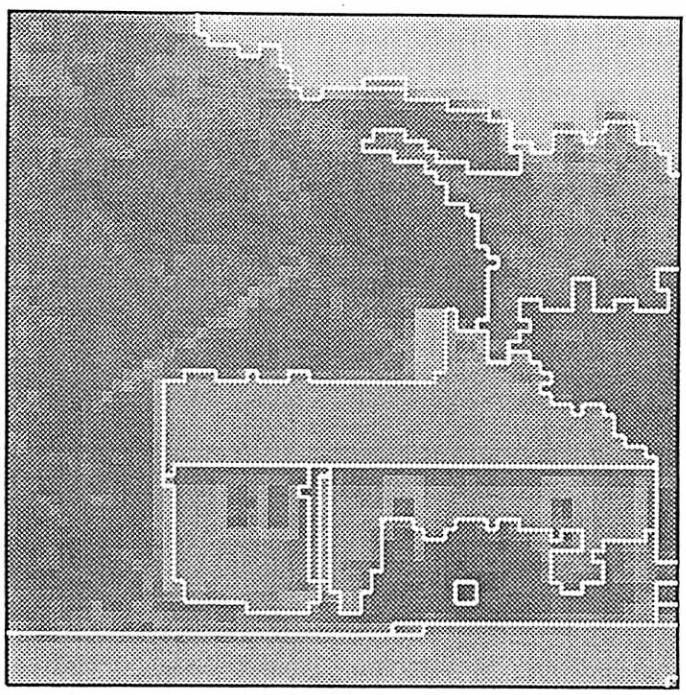


Figure 6.66: Barn in Segmentation of Road 16

the image characteristics of the bush area and grass area are quite similar, but an interpretation process that noted that an area labeled "grass" extended above the horizon might post a region–segmentation goal with the `object-labels` constraint of '(grass) so that any portions of the original region that did not represent grass would form separate regions. Although the resegmentation of Figure 6.72 has not completely discriminated all areas of bush (the highlighted portion of the bush is practically identical to the grass), it does provide enough evidence for a bush that the interpretation process could then use the line–segmentation schema to partition the remaining grass region according to the hypothesized horizon (Figures 6.73 and 6.74).

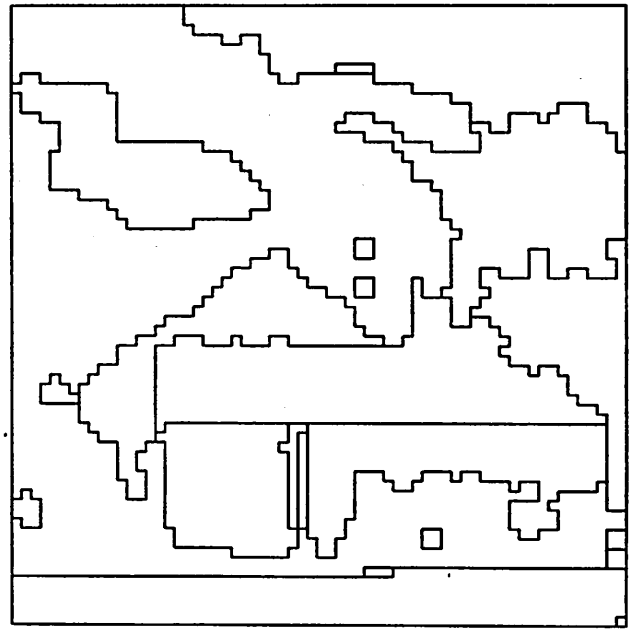


Figure 6.67: Barn in Resegmentation of Road 16

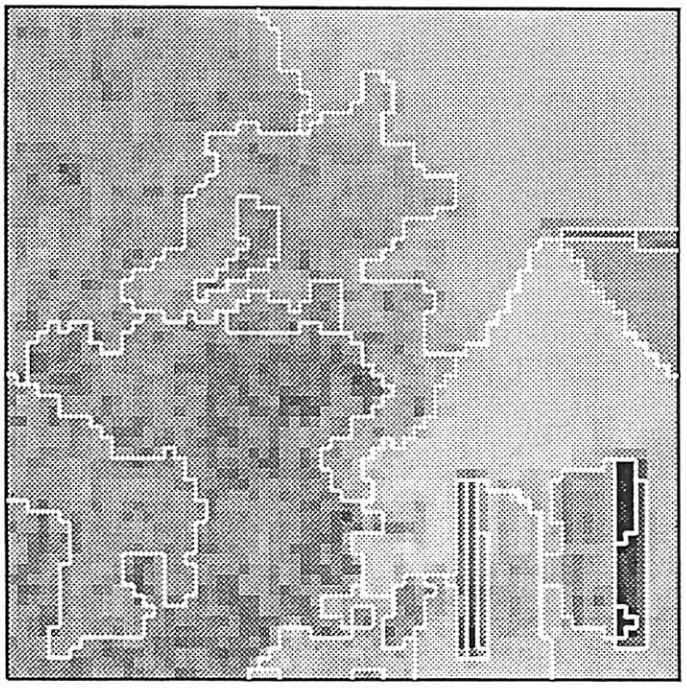


Figure 6.68: Image Data for Region Containing Tree and Sky

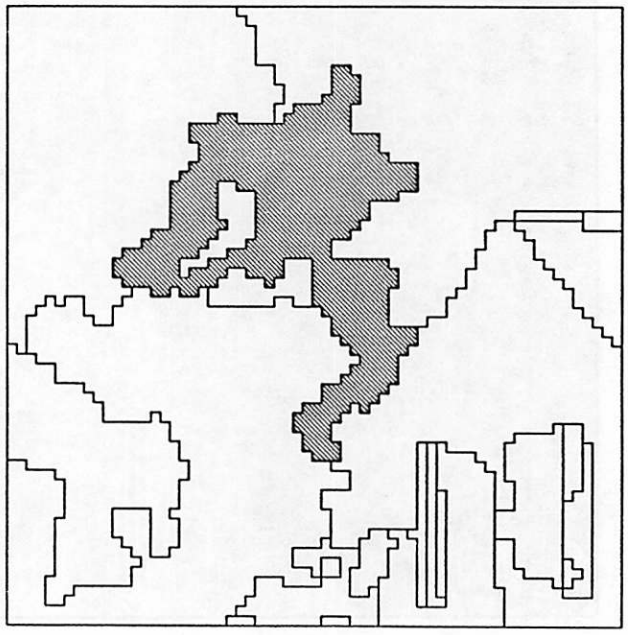


Figure 6.69: Tree/Sky Region for Resegmentation

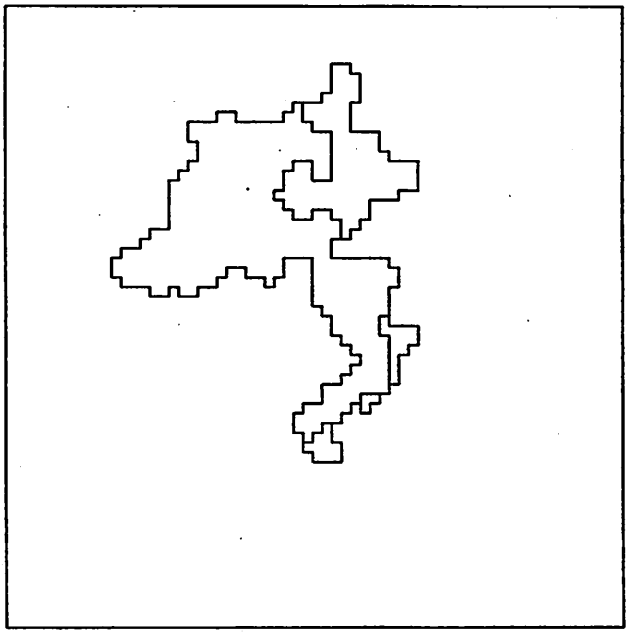


Figure 6.70: Tree and Sky Regions

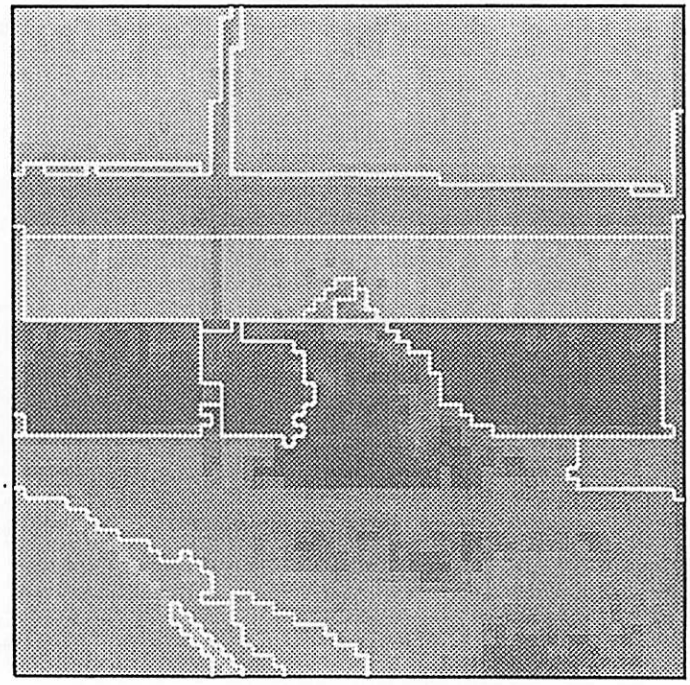


Figure 6.71: Bush/Grass Region

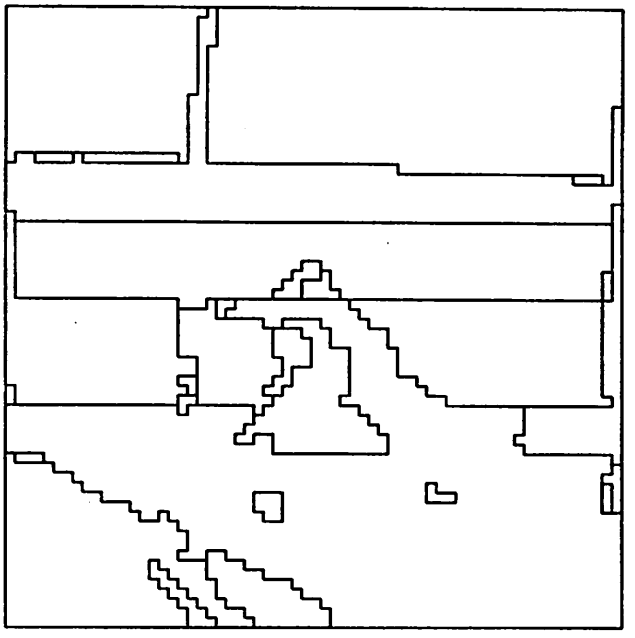


Figure 6.72: Resegmentation of Bush/Grass Region

Another way in which expectation–driven processing could be used to aid the interpretation process would be in situations where it was not necessary to completely label the entire image, but only necessary to provide an approximate description of the scene. In such situations, it might be desirable for the interpretation process to first request an inexpensive low–resolution initial segmentation from the IDDS schema, and then request resegmentation only for those areas of the image that could not be identified. Figure 6.75 shows the low–resolution segmentation that was produced from the House 15 image (Figure 6.1). In this segmentation there are two large areas that correspond to multiple scene objects. The first is the region representing roof and house wall, and the second is the region representing bush and grass. However, given context from an initial interpretation of the low–resolution segmentation it would be quite possible for the interpretation system to hypothesize "house" (i.e. house–wall, roof, and shutter) for the former region, and "foliage" (i.e. grass, bush, and tree) for the latter.

Given these hypotheses, it then becomes possible to post goals for resegmentation of these regions using this semantic information. The segmentation of the house region produced by using the `object-labels` constraint of '(house–wall shutter roof) generated the set of regions shown in Figure 6.76, and the goal for region–segmentation of the foliage region using the `object-labels` constraint of '(grass bush tree) resulted in the set of region tokens shown in Figure 6.77. When the regions of these two resegmentations are combined with the original top–level segmentation data (Figure 6.78), the resulting intermediate–level representation forms a reasonably good description of scene content.

6.3.2 Extraction of Shape Constrained Tokens

A different form of processing that may be required of the intermediate–level schemas is to produce tokens that have specific shape, but indeterminate location. In these situations, the constraints on the goal specify the desired shape

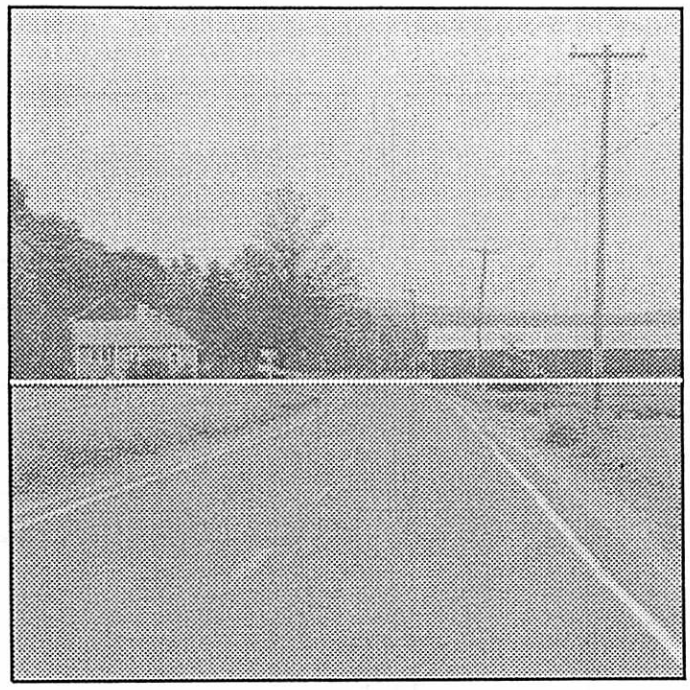


Figure 6.73: Hypothesized Horizon Line

characteristics of the desired tokens, but little or no information about the specific location. For example, if there has been an inference of the presence of a pair of shutters on the side of the house in Figure 6.80, an interpretation process might request a resegmentation of the area defined by that pair in an attempt to define tokens that could be unambiguously labeled as either window or shutter. The goal specification for this request, shown in Figure 6.79, includes an evaluation-constraint that indicates a preference for region tokens that are rectangular.

Figure 6.81 shows the result of this goal specification over the three areas bounded by pairs of initially hypothesized shutters in the original segmentation (Figure 6.80). The dark lines in this figure represent region boundaries that were constructed by GOLDIE in response to the definition of an "area of interest" by the interpretation process. Each of these newly defined regions was then resegmented according to the specified goal. The shape-evaluation function used in the `evaluation-constraint` in this goal computes the degree of overlap between

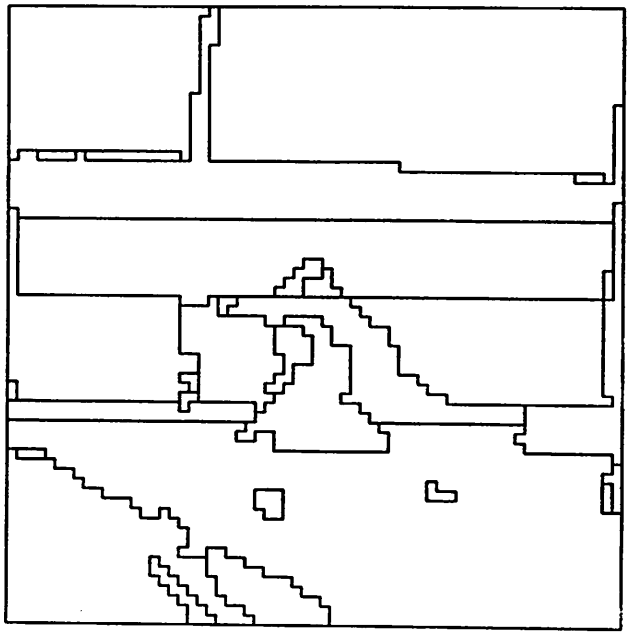


Figure 6.74: Line–Segmentation of Grass Region

each of the segmentation regions and the bounding rectangle for the region, thus computing the rectangularity of the region. Using the size of each region in the segmentation as a weight, the weighted sum of these rectangularity values is used to produce the evaluation score. Note that although image noise, aliasing, and window reflections prevent a "perfect" segmentation of these areas, the resegmentation process has provided a set of tokens that exhibit the desired rectangular characteristics and are appropriate for interpretation of windows and shutters.

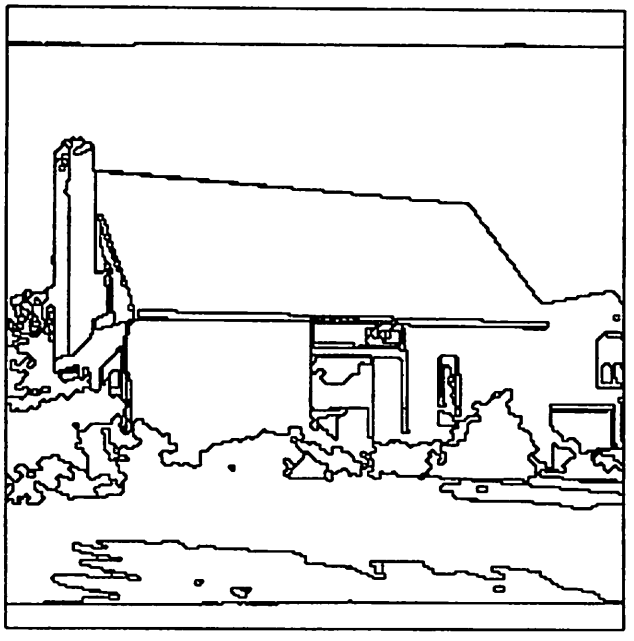


Figure 6.75: IDDS Segmentation With Low–Resolution Constraint

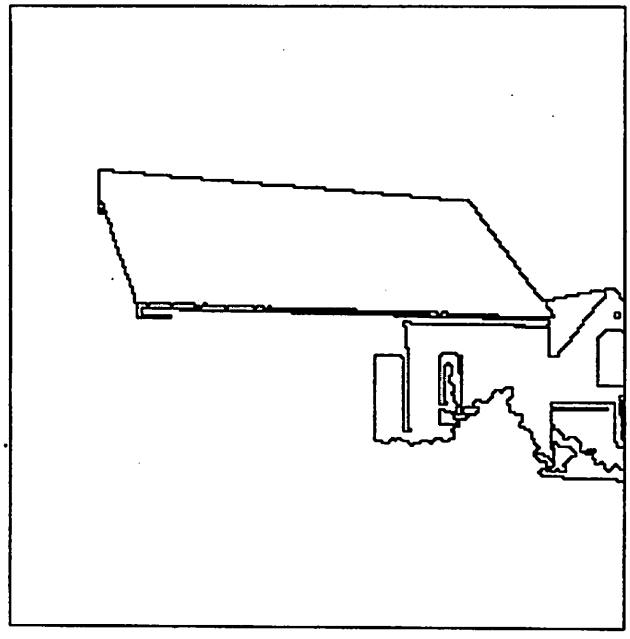


Figure 6.76: Resegmentation of House Region

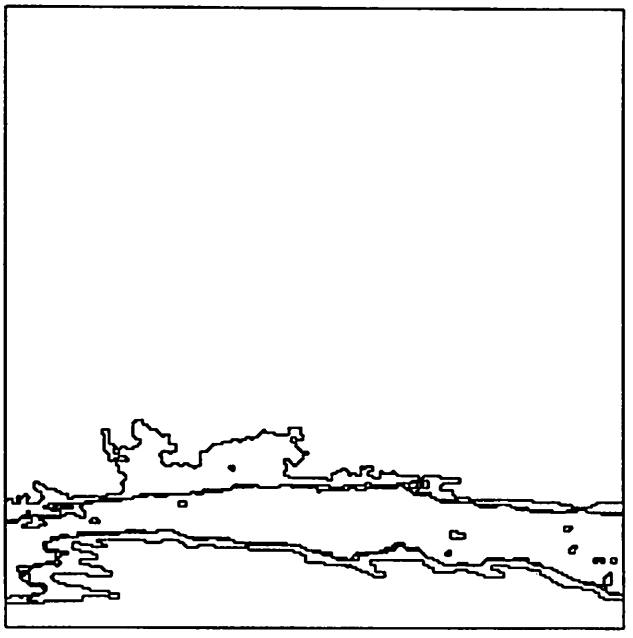


Figure 6.77: Resegmentation of Foliage Region

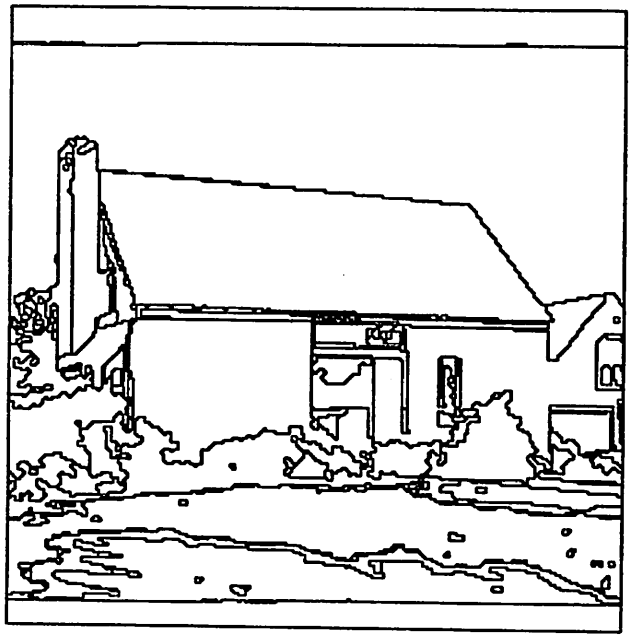


Figure 6.78: Resegmentation of House 15 Image

Region-Segmentation
'((*region–token* . Region-000042)
(*evaluation–constraint* . ((shape-evaluation 'rectangular) 1.0))
(*object–labels* . (shutter))
(*region-characteristic* . smooth))

Figure 6.79: Goal for Resegmentation of Shutters

6.3.3 Extraction of Obscure Image Tokens

In still other cases, factors such as noise, aliasing, or occlusion may prevent the extraction of image tokens corresponding to objects that have been hypothesized by interpretation processes. These hypotheses could be "hallucinated" (i.e. proposed by high–level processes on the basis of relational constraints without any direct evidence in the image), or they could be based on fragmentary or incomplete evidence. In these situations, it is possible for the intermediate–level schemas of GOLDIE to operate as "assistants" to the high–level processes in an attempt to extract the desired tokens.

A simple example of this type of extraction process was the discrimination of house wall and sky that was shown in the description of the IDDS schema in Chapter 5 (Figure 5.7). The characteristics of the image data in this instance were such that none of the region segmentation algorithms available to the system were able to identify the boundary between these two visually distinct regions. However, through the insertion of an extended version of a line that had been extracted by the line extraction schema, the boundary was enforced upon the ambiguous image data. In this particular example, the non–semantic evidence for the existence of two distinct regions was strong (the dividing line was long, reasonably high contrast, and needed to be extended by only a few pixels), so that the intermediate–level IDDS schema was able to utilize the line–segmentation schema directly to perform the operation. In other cases, however, evidence derived from the semantic information provided by the interpretation schemas may be necessary to establish the probable existence of the the object(s) in question. In situations of this type, the high level process has been able to determine that a region token with specific location and characteristics should exist at a particular location, but it remains for the intermediate–level process to determine whether or not the image data supports or contradicts the evidence.

This type of processing requires close interaction between the schemas of the high and intermediate–level schemas. As an example, consider the extraction of

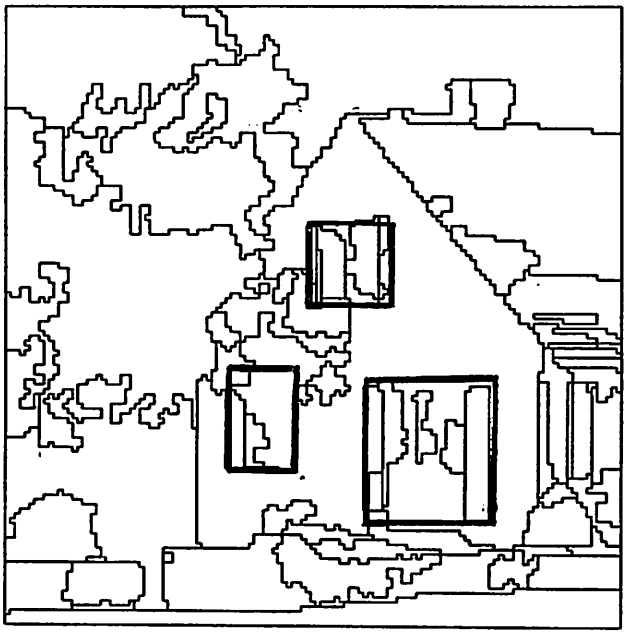


Figure 6.80: IDDS Segmentation of Portion of House 1

the telephone pole (Figure 6.82) that was missing from the IDDS segmentation of the Road 16 image in Figure 6.17. It is quite understandable why the vertical portion this pole was not completely extracted in the original segmentation. The image characteristics of the central area of the pole are so similar to those of the background foliage that an image or intermediate–level process is not able to distinguish the two objects. However, it is possible that a high–level process could predict the general location of the pole from the position of the foreground pole and the road. The line tokens in this general area (Figure 6.83) could the be examined for a set of parallel lines that might be related to the pole (Figure 6.84). At this point, the high–level process could post a goal for line–extraction over the set of region tokens that intersect these hypothesized pole lines to extract any low–contrast lines that had been thresholded out in the original line–extraction process. Given this new set of lines (Figure 6.85), the high–level process would be able to find an additional line to support the existence of a telephone pole at

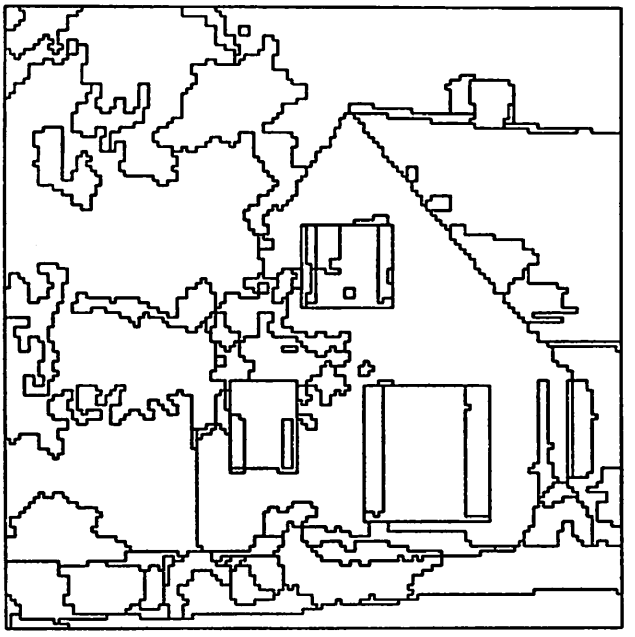


Figure 6.81: Rectangular Shutter Regions

the hypothesized location (Figure 6.86). When the line–segmentation schema is presented with this set of lines as a goal constraint for direct line insertion, this schema produces the region shown in Figure 6.87, both confirming the existence of the pole at the hypothesized location and providing a new region token that can be labeled by the interpretation process.

A different type of interaction between the high and intermediate–level schemas could take place during the verification of the existence of telephone lines between these two poles. The actual representation of these lines in the image data is a very thin chain of pixels that are only slightly darker that the surrounding sky region, so it is understandable that there are no regions corresponding to these lines in the segmentation. If a high–level process attempts to verify the existence of the lines, it can post the region–segmentation goal shown in Figure 6.88.

Since the `segmentation-resolution` is specified as very–high, and since the

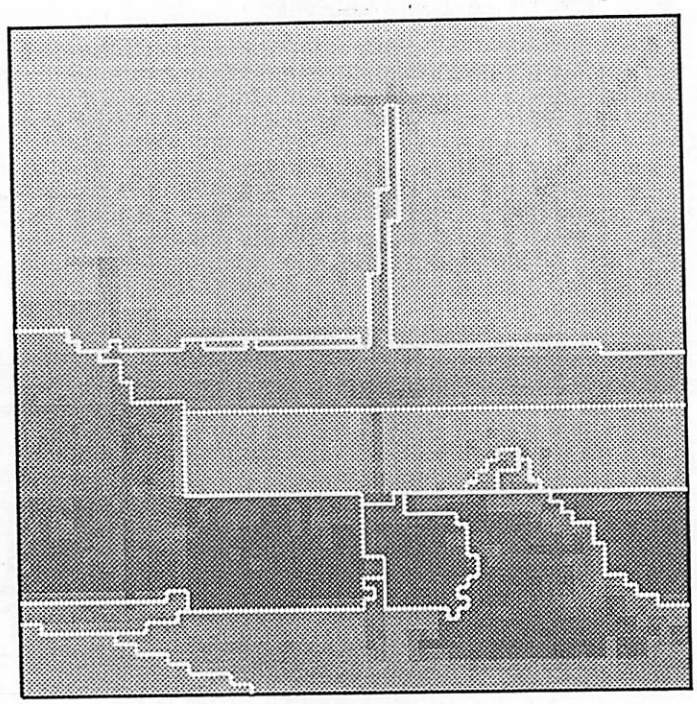


Figure 6.82: Telephone Pole

`evaluation-constraint` is specified as nil, the region–segmentation schema returns the segmentation produced by the zero–crossing process on the Intensity image feature at a sensitivity setting of 1.0 without evaluating this segmentation for overfragmentation. As seen in Figure 6.89, this segmentation, while dramatically overfragmented, does contain regions that correspond to wires. If the high–level schema were able to identify these as such, perhaps due to shape, orientation, and location, one of these regions could be specified as the `exemplar-region` constraint to a second invocation of the region–segmentation schema. In this second invocation, the region–segmentation schema instance is able to identify the particular image feature in which the exemplar region differs the most from the background sky region (Blue), and is able to set thresholds that permit the extraction of the wire regions shown in Figure 6.90. Note that the extraction process is quite specific to the particular characteristics of the exemplar region. Even though there is evidence for another set of wires at the top of the pole

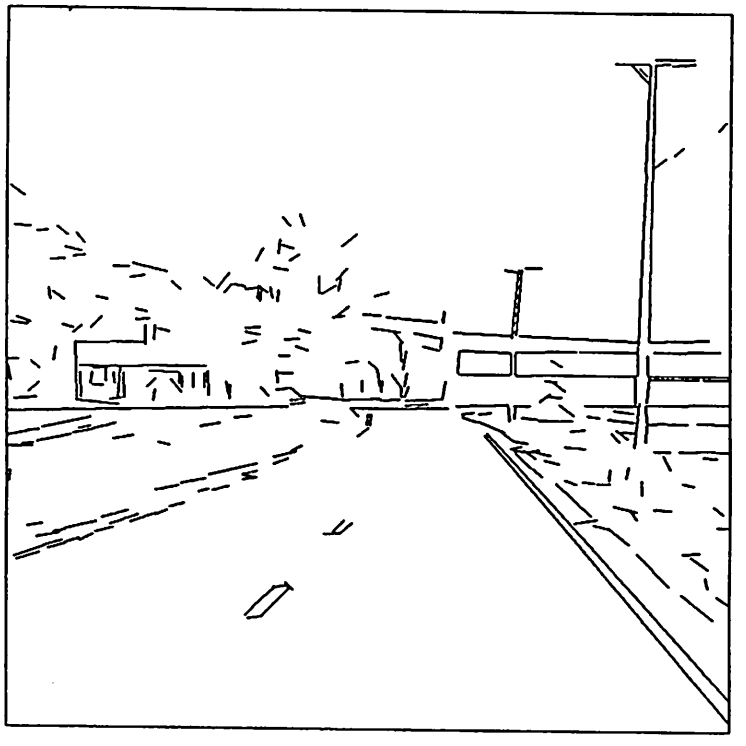


Figure 6.83: Line Tokens for Road 16 Image

in Figure 6.89, these wires have not been extracted in the region segmentation process because the image characteristics of those wires are somewhat different from the characteristics of the wires in the exemplar region[3].

This form of interaction between high and intermediate–level schemas is complex, and would potentially require high–level interpretation schemas that contained a great deal of knowledge about the nature of intermediate–level processing. However, without the concept of intermediate–level control, each of the object–based schemas would be required to understand all of the low and intermediate–level processes that could potentially be applicable to its own particular set of problem situations. Thus, although the level of interaction proposed in these examples is complex, it is also modular. If an interpretation schema can

[3]Due to view angle, distance, or some other factor, the mean Blue value for the upper set of wires is about 10 graylevels higher (out of a range of 256) than that of the lower set of wires.

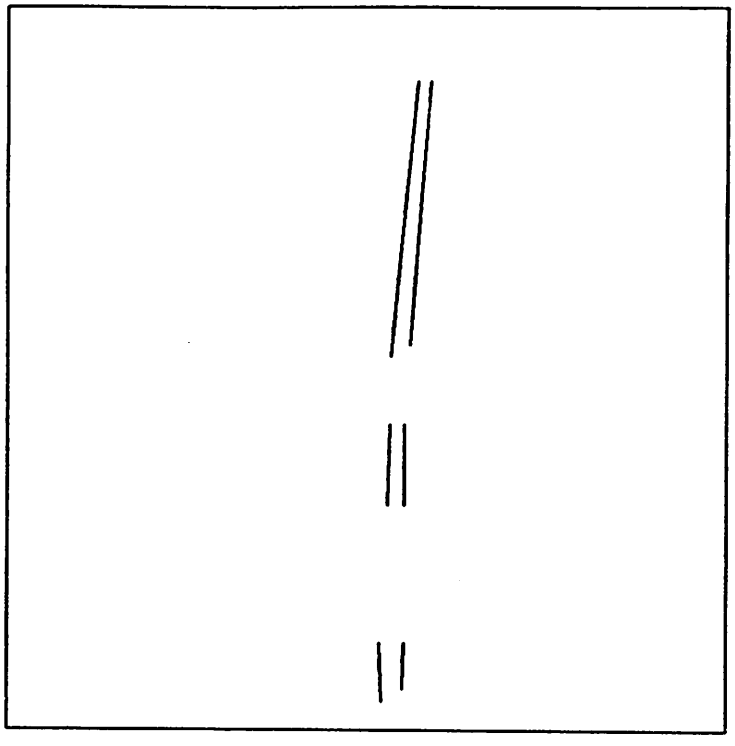


Figure 6.84: Parallel Lines Near Hypothesized Telephone Pole

be given some form of knowledge about intermediate–level processing (e.g. *"an exemplar region for certain types of small objects can potentially be found in highly overfragmented segmentations"*), the intermediate–level schemas may be used to produce this data.

6.4 Conclusion

The experiments presented in this chapter have demonstrated both the utility of the goal–directed control paradigm of GOLDIE and the way in which intermediate–level knowledge may be used to achieve the goals of the image interpretation process. The mechanisms embodied in the design of this system allow the creation of image tokens that serve, rather than direct, the image interpretation process. When operating in a data–directed mode, the system is able to create tokens that enable the creation of reasonable semantic hypotheses, and

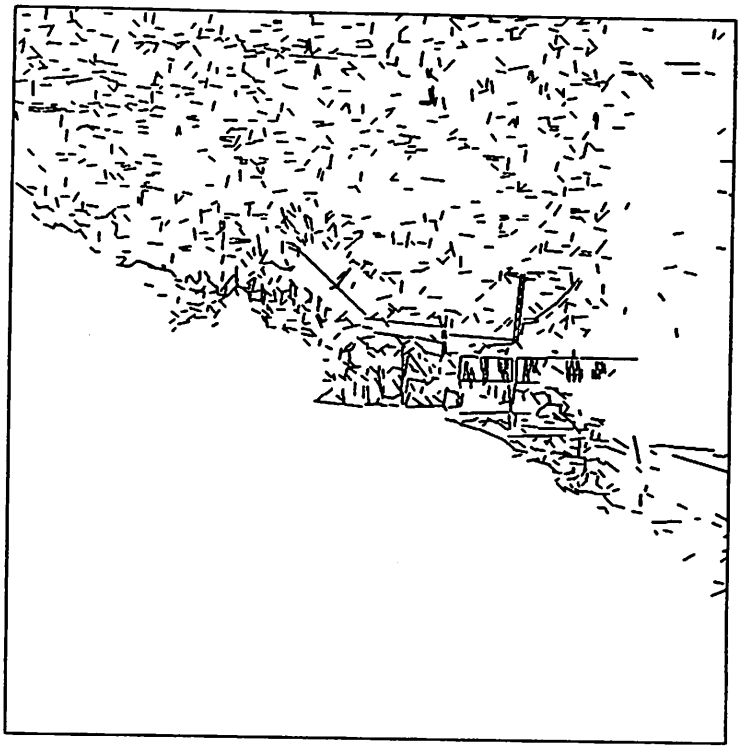


Figure 6.85: Additional Lines for Road 16 Image

when operating in the expectation–driven mode, the system is able to examine the data to verify the existence of semantic objects whose existence is ambiguous.

It should be emphasized that although we have simulated the expectation–driven goals to these examples, this does not mean that we have presented some very special case solutions to a random set of problems. Instead, what we have done is to take a set of intermediate–level schemas that were constructed to operate in a data–driven environment, and then show how the goal–directed control mechanisms of GOLDIE enable these same schemas to operate in an expectation–driven environment. We have shown a number of examples in which an interpretation process could specify a set of goal constraints in very high–level terms that enable the schemas of GOLDIE to produce the desired data.

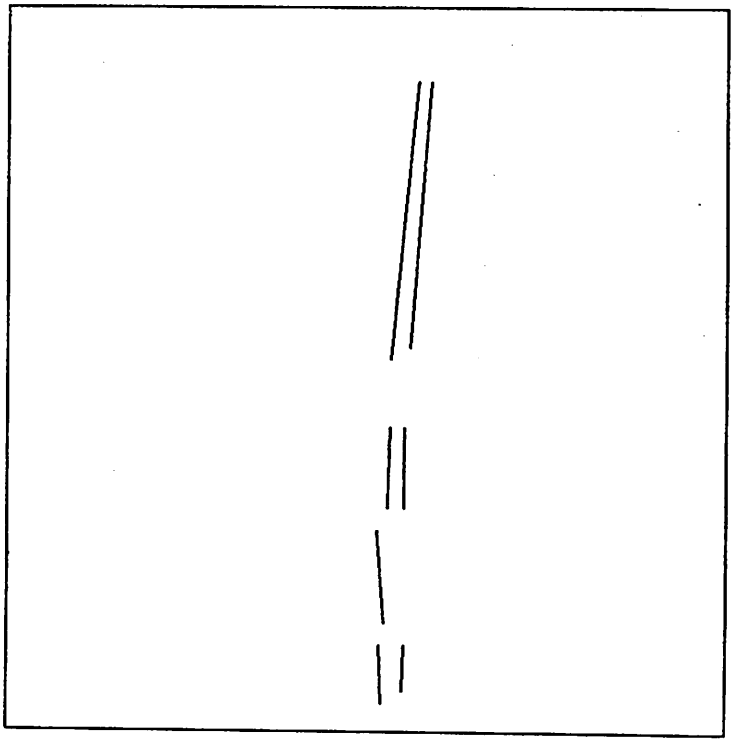


Figure 6.86: New Set of Parallel Lines Near Hypothesized Telephone Pole

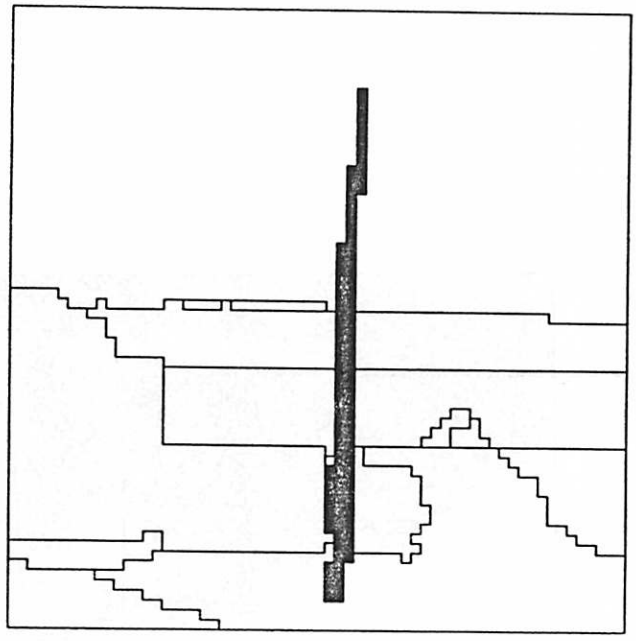


Figure 6.87: Region Token for Telephone Pole

Region-Segmentation
'((*region–token* . Region-00492)
(*region–characteristic* . smooth)
(*segmentation-resolution* . very-high)
(*evaluation-constraint* . nil)
(*object–labels* . (wire))

Figure 6.88: Goal for High–Resolution Segmentation of Sky Region

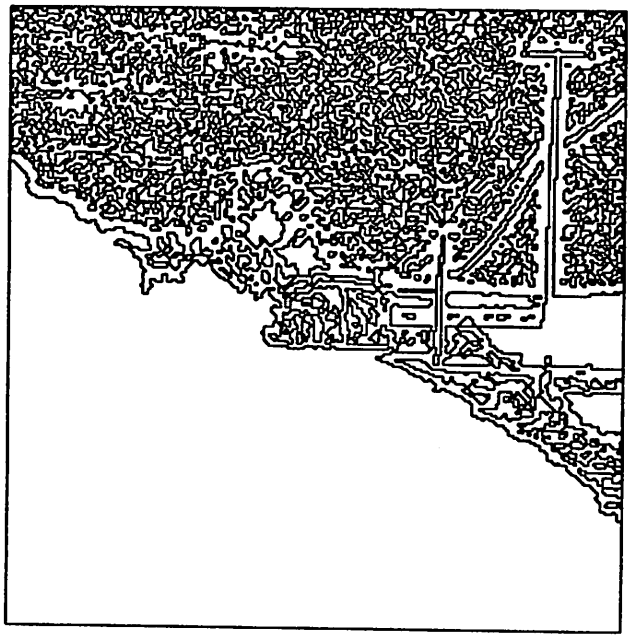


Figure 6.89: Oversegmentation of Sky Region

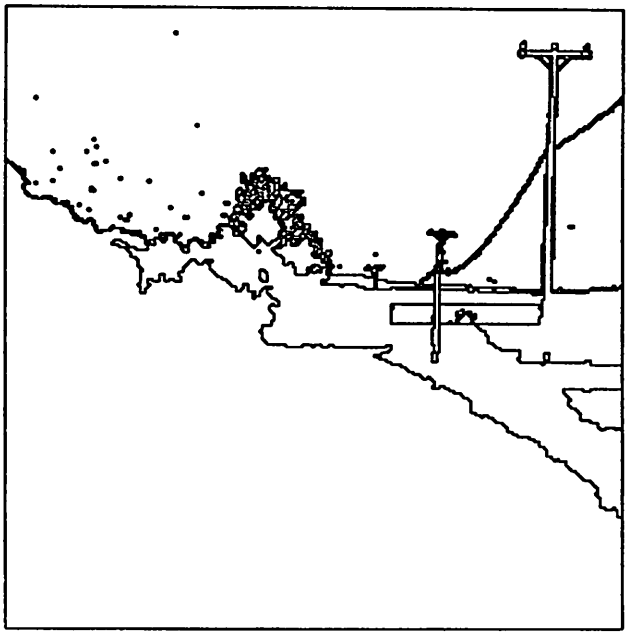


Figure 6.90: Telephone Wire Regions

Chapter 7

Contributions and Future Research

7.1 Contributions

The system described in this dissertation presents a new approach to the image interpretation process. Image interpretation is not a linear process in which an image is first segmented and then interpreted; rather it is a complex process in which a variety of low, intermediate, and high–level processes dynamically interact to produce an interpretation of the image.

The primary concept of this system is that of a *goal*; an expression that describes the characteristics of the low and intermediate–level data that would facilitate the interpretation process. The use of this goal–based model has been made possible through the construction of:

- structures for the representation of explicit knowledge of the image domain,
- a single flexible, extensible, and modular representation for intermediate–level data, and
- control structures that enable interaction between the various strategies and techniques at all three levels of the interpretation process.

7.1.1 Knowledge of the Image Domain

The interpretation of an image involves at least three different mappings. The first occurs when a three–dimensional scene is mapped into a discrete two–dimensional digital representation of light intensities in computer memory. The

second occurs when these discrete data are mapped into a variety of image abstractions, and the third occurs as these intermediate–level abstractions (tokens) are mapped into a description of the original scene. Traditional approaches to the interpretation process have used explicit representation of knowledge only at the level of the final mapping between tokens and object descriptions.

In GOLDIE, knowledge is represented explicitly at all three levels of this mapping. Knowledge about the characteristics of image data is expressed both procedurally, in the expression of the various image transforms and segmentation algorithms, and declaratively, in the representation of the specific image features and segmentation algorithms that are appropriate to specific goals. The knowledge about tokens is expressed in the data–oriented rules that are used to evaluate region tokens for resegmentation or merging. The rules themselves are procedural, while the specification of the specific set of rules to use in a particular situation is declarative.

The knowledge of the semantic domain is currently confined to the set of semantic evaluation rules in GOLDIE. However, since this system is designed to interface with a high–level interpretation system, the full power of such a system is implicitly a part of the overall design.

7.1.2 Intermediate–Level Data Representation

By providing a consistent intermediate–level representation of data, GOLDIE provides a mechanism through which the processes at the various levels of the system may communicate. Image data, regions, lines, and hypotheses are all represented in a single consistent structure that can be viewed as the *blackboard* of the system. Once information becomes "public" (i.e. is available for use by other processes of the system) it is stored in this representation regardless of whether it is information about the existence of a token, measurements of token features, hypotheses about the "quality" of the token, or hypotheses about the semantic labels which can be applied to the token.

With this representation, the sharing of data becomes trivial. The name of a token provides access to all public information about that token. Rather than representing this data in a variety of structures such as region label planes and feature arrays, this symbolic representation allows explicit access to token data from any point in the system.

7.1.3 Control

The use of the *schema*, the explicit representation of the various strategies which may be employed to satisfy a goal, as the control structure at the intermediate level allows GOLDIE to become an integral component of the interpretation process. The schemas allow a variety of strategies for goal satisfaction to be employed in the production of the desired data. Hypothesize–and–test can be used as a control paradigm, and the low level processes of the system, which in themselves have no knowledge of the interpretation domain, can be used to produce the intermediate–level data that is tuned to the expectations of the interpretation processes.

7.2 Future Research

The structure of GOLDIE, as it is currently implemented, provides a new tool for the production of high quality image segmentations. Next, it will be necessary to integrate this system with the high–level interpretation system in VISIONS. Although there are no conceptual difficulties with this process, two separate issues must be addressed before the integration is complete. Initially, a variety of hardware and multiprocessing difficulties must be resolved in order to allow GOLDIE and the interpretation processes to run in a closely coupled environment. More importantly, however, the conceptual approach to the design of specific interpretation schemas must change. Previous implementations of interpretation schemas have been designed under the assumption that the goal

was to assign an interpretation to a specific predefined set of tokens. Now that tokens have become dynamic, schemas for interpretation may be designed with considerably more power. Thus, new designs for the interpretation schemas will need to be created to make use of that power.

A second area of research has to do with the intermediate–level evaluation of token data and control of low–level processes. The current model for search in the hypothesis–and–test paradigm is breadth–first. Given a set of goal constraints, the schemas of GOLDIE construct an ordered set of process specifications and then invoke these processes sequentially until the evaluations functions indicate that acceptable data has been produced. Future implementations of the system will make this search process more efficient by providing "failure analysis". By providing evaluation functions that determine why certain data is not acceptable, we may be able to choose a more profitable alternate strategy for the production of the desired data.

Other issues for future research involve low and intermediate processes local to GOLDIE itself. Given the nature of this system as an extensible, goal–directed mechanism for the control of low–level processes, the system is by definition incomplete; there will always be additional low or intermediate level processes and schemas which may be added to the system. New types of token, such as surface, ribbon, or area tokens, as well as the low–level processes which create theses tokens, could easily be added to the representation.

New forms of two–dimensional sensory data such as multispectral scanner, tactile, range, or x–ray data could be integrated into the image structure. Additional image domains could could be explored, leading to the creation of new classes of image tokens such as soft or hard tissue, as well as new sets of rules to evaluate these token classes.

Temporal sequences of image data, such as sequential frames from a moving sensor, will also provide a new image domain. With the addition of token relations that indicate that a token in one image is related to a similar token in another

image, the system will be able to describe the way in which objects move in time. Once an environmental model has been constructed, the tokens corresponding to a previous frame could be translated according to the flow equations to provide a new set of tokens for the current frame purely at the intermediate level. The evaluation functions of GOLDIE could then be used to direct low–level processing strictly to those image areas where the image data did not match the prediction. Similar mechanisms could be used in stereo imagery to correlate events from a pair of images.

The list could go on; since GOLDIE is a general system for goal–directed processing, any interpretation task that makes use of intermediate–level tokens could conceivably make use of the mechanisms provided through this system.

7.3 Conclusions

Even though GOLDIE has not as yet been fully integrated with a high–level interpretation system, the experimental data presented in Chapter 6 clearly demonstrates the value of this approach. When the schemas have the ability to direct the creation or modification of intermediate–level data, it becomes possible to reduce the complexity and uncertainty of the interpretation process.

Even without integration between the schemas of GOLDIE and those of the interpretation process, the value of the *goal–directed* approach may be seen in the quality of the segmentations produced by the initialization schema. The goal–directed control, modularity, and extensibility of this system provide the foundation for a variety of new approaches to image interpretation.

Bibliography

[1] P. Anandan. *Measuring Visual Motion from Image Sequences.* Ph.D. Dissertation, Computer and Information Science Department, University of Massachusetts, Amherst, Massachusetts 01003, 1987. Also as Technical Report 87-21 Computer and Information Science Department, University of Massachusetts.

[2] P. Anandan. *Motion and Stereopsis.* Technical Report 85-52, Computer and Information Science Department, University of Massachusetts, Amherst, Massachusetts 01003, December 1985.

[3] M. A. Arbib. Automata theory in the context of theoretical neurophysiology. In R. Rosen, editor, *Textbook of Theoretical Biology*, Academic Press, New York, NY, 1972.

[4] Michael A. Arbib. Segmentation, schemas, and cooperative communication. In S. Leven, editor, *Studies in Mathematical Biology, Part 1*, pages 118–155, MAA Studies in Mathematics, 1978.

[5] Michael A. Arbib and Edward M. Riseman. *Computational Techniques in Visual Systems - Part 2.* Technical Report 76-10, Computer and Information Science Department, University of Massachusetts, Amherst, Massachusetts 01003, July 1976.

[6] F. Atteneave. Informational aspects of visual perception. *Psych. Rev.*, 61:183–193, 1954.

[7] D. H. Ballard. *Hierarchical Recognition of Tumors in Chest Radiographs.* Basel: Birkhauser-Verlag (ISR-16), 1976.

[8] D. H. Ballard, C. M. Brown, and J. A. Feldman. An approach to knowledge-directed image analysis. In Allen R. Hanson and Edward M. Riseman, editors, *Computer Vision Systems*, pages 271–282, Academic Press Inc., New York, NY, 1978.

[9] D. H. Ballard, M. Marinucci, F. Proietti–Orlandi, A. Rossi-Mari, and L. Tentari. Automatic analysis of human haemoglobin fingerprints. In *Proceedings, 3rd Meeting, International Society of Haemotology*, London, August 1985.

[10] Dana H. Ballard and Christopher M. Brown. *Computer Vision.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey 07632, 1982.

[11] H. B. Barlow. Sensory mechanisms, the reduction of redundancy, and intelligence. In K. N. Leibovic, editor, *Information Processing in the Nervous System*, pages 209–230, Springer-Verlag, New York, NY, 1969.

[12] Avron Barr and Edward A. Feigenbaum, editors. *The Handbook of Artificial Intelligence Vol. I.* William Kaufmann Inc., Los Altos, CA, 1981.

[13] Harry G. Barrow and J. Martin Tenenbaum. Recovering intrinsic scene characteristics from images. In Allen R. Hanson and Edward M. Riseman, editors, *Computer Vision Systems*, pages 3–26, Academic Press Inc., New York, NY, 1978.

[14] Harry G. Barrow and Jay M. Tenenbaum. *MSYS: A System for Reasoning About Scenes.* Technical Report 121, SRI International, 333 Ravenswood Ave., Melno Park, CA 94025, April 1976.

[15] R. Belknap, E. Riseman, and A. Hanson. The information fusion problem and rule–based hypotheses applied to complex aggregations of image events. In *Proceedings: IEEE–CVPR Conference*, pages 227–234, June 1986.

[16] R. Belknap, E. Riseman, and A. Hanson. *The Information Fusion Problem and Rule–Based Hypotheses Applied to Complex Aggregates of Image Events.* Technical Report 87-48, Computer and Information Science Department, University of Massachusetts, Amherst, Massachusetts 01003, 1987.

[17] Valdis Berzins. Accuracy of Laplacian edge detectors. *Computer Vision, Graphics, and Image Processing*, 27:195–210, 1984.

[18] J. Ross Beveridge, Joey Griffith, Ralf R. Kohler, Allen R. Hanson, and Edward M. Riseman. *Segmenting Images Using Localized Histograms and Region Merging.* Technical Report 87-88, Computer and Information Science Department, University of Massachusetts, Amherst, Massachusetts 01003, 1987.

[19] S. Bharwani, R. Arkin, E. Riseman, and A. Hanson. Visual depth recoveryfor obstacle avoidance in a mobile robot. In *Proceedings of the IEEE Industrial Electronics Society, Robotics: Trends, Technology and Applications Conference*, Madrid, Spain, September 1987.

[20] S. Bharwani, E. Riseman, and A. Hanson. Refinement of environmental depth maps over multiple frames. In *Proceedings of the IEEE Workshop on Motion: Representation and Analysis*, Charleston, SC, May 1986.

[21] Michael Boldt and Richard Weiss. *Token-Based Extraction of Straight Lines.* Technical Report 87-104, Computer and Information Science Department, University of Massachusetts, Amherst, Massachusetts 01003, 1987.

[22] R. C. Bolles. *Verification Vision within a Programmable Assembly System.* Technical Report AIM-295, Aritifical Intelligence Laboratory, Stanford University, Palo Alto, CA, 1976.

[23] Ronald J. Brachman. What is-a is and isn't: an analysis of taxonomic links in semantic networks. *Computer*, 16(10):30–36, October 1983.

[24] Rafael Bracho and Arthur C. Sanderson. Segmentation of images based on intensity gradient information. In *Proceedings: IEEE–CVPR Conference*, pages 341–347, 1985.

[25] J. E. Bresenham. Algorithm for computer control of digital plotter. *IBM Systems Journal*, 4(1):25–30, 1965.

[26] C. R. Brice and C. L. Fennema. Scene analysis using regions. *Artificial Intelligence*, 1:205–226, 1970.

[27] R. Brooks. Symbolic reasoning among 3–dimensional models and 2–dimensional images. *Artificial Intelligence*, 17:285–349, 1981.

[28] R. A. Brooks. Model based three–dimensional interpretation of two dimensional scenes. In *Proceedings: Seventh International Joint Conference on Artificial Intelligence*, pages 619–624, 1981.

[29] R. A. Brooks, R. Griener, and T. O. Binford. A model–based vision system. In *Proceedings: DARPA Image Understanding Workshop*, pages 36–44, May 1978.

[30] Rodney A. Brooks and Thomas O. Binford. Interpretive vision and restriction graphs. In *Proceedings of The First Annual National Conference on Artificial Intelligence*, pages 21–27, August 1980.

[31] N. A. Bryant. Integration of socioeconomic data and remotely sensed imagery for land use applications. In *Proceedings of Caltech/JPL Conference on Image Processing Technology, Data Sources and Software for Commercial and Scientific Applications*, pages 91–98, Pasadena CA, November 1976.

[32] D. Brzakovic and J. T. Tou. Boundary determination of object surfaces via textural information. In *Seventh International Conference on Pattern Recognition*, pages 280–283, July 1984.

[33] J. B. Burns, A. R. Hanson, and E. M. Riseman. Extracting straight lines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(4):425–455, July 1986.

[34] J. B. Burns, A. R. Hanson, and E. M. Riseman. Extracting straight lines. In *Proceedings Seventh International Conference on Pattern Recognition*, pages 482–485, Montreal, July 1984.

[35] Peter Burt. Pyramid–based extraction of local image features with applications to motion and texture analysis. In *Proceedings 1982 SPIE Conference on Robotics and Industrial Inspection Systems*, 1982.

[36] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, November 1986.

[37] John F. Canny. *Finding Edges and Lines in Images.* Technical Report 720, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, June 1983.

[38] P. C. Chen and T. Pavlidis. Image segmentation as an estimation problem. *Computer Graphics and Image Processing*, 12:153–172, 1980.

[39] M. B. Clowes. On seeing things. *Artificial Intelligence*, 2:79–116, 1971.

[40] F. S. Cohen, D. B. Cooper, J. F. Silverman, and E. B. Hinkle. Simple parallel hierarchical and relaxation algorithms for segmenting textured images based on noncausal Markovian random field models. In *Seventh International Conference on Pattern Recognition*, pages 1104–1107, July 1984.

[41] Paul R. Cohen. *Numeric and Symbolic Reasoning about Uncertainty in Expert Systems.* Technical Report 85-25, Computer and Information Science Department, University of Massachusetts, Amherst, Massachusetts 01003, 1985.

[42] Paul R. Cohen and Edward A. Feigenbaum, editors. *The Handbook of Artificial Intelligence Vol. III.* William Kaufmann Inc., Los Altos, CA, 1982.

[43] Guy B. Coleman. *Image Segmentation by Clustering.* Technical Report 750, Image Processing Institute, University of Southern California, Los Angeles, CA 90007, July 1977.

[44] James L. Crowley. *A Representation for Visual Information.* Technical Report CMU-RI-TR-82-7, The Robotics Institute Carnegie-Mellon University, Pittsburgh, Pennsylvania 15213, November 1982.

[45] Larry S. Davis. *Cooperating Processes for Low-Level Vision: A Survey.* Technical Report TR-851, Computer Vision Laboratory University of Maryland, College Park, Maryland 20742, January 1980.

[46] Larry S. Davis, Ludvik Janos, and Stanley Dunn. *Efficient Recovery of Shape from Texture.* Technical Report TR-1133, Computer Vision Laboratory University of Maryland, College Park, Maryland 20742, April 1982.

[47] Larry S. Davis and Azriel Rosenfeld. Hierarchical relaxation for waveform parsing. In Allen R. Hanson and Edward M. Riseman, editors, *Computer Vision Systems*, pages 101–110, Academic Press Inc., New York, NY, 1978.

[48] D.L.Milgram. Region extraction using convergent evidence. *Computer Graphics and Image Processing*, 11:1–12, 1979.

[49] B. Draper, R. Collins, J. Brolio, J. Griffith, A. Hanson, and E. Riseman. Tools and experiments in knowledge–based interpretation of road scenes. In *Proceedings: DARPA Image Understanding Workshop*, pages 178–193, Morgan Kaufman, February 1987.

[50] B. Draper, A. Hanson, and E. Riseman. *A Software Environment for High Level Vision.* Technical Report, Computer and Information Science Department, University of Massachusetts, 1988. In preparation.

[51] B. A. Draper, R. T. Collins, J. Brolio, A. R. Hanson, and E. M. Riseman. The schema system. Technical Report, Computer and Information Science Department, University of Massachusetts, 1988. In preparation.

[52] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis.* John Wiley and Sons, New York, NY, 1973.

[53] S. A. Dudani and A. L. Luk. Locating straight–line edge segments on outdoor scenes. In *Proceedings Pattern Recognition and Image Processing*, pages 367–377, 1977.

[54] R. W. Ehrich and J. P. Foith. Topology and semantics of intensity arrays. In Allen R. Hanson and Edward M. Riseman, editors, *Computer Vision Systems*, pages 111–127, Academic Press Inc., New York, NY, 1978.

[55] Paul Eichel and Edward Delp. Sequential edge extraction in correlated random fields. In *Proceedings: IEEE–CVPR Conference*, pages 14–21, 1985.

[56] L. Erman, F. Hayes-Roth, V. Lesser, and D. Reddy. The Hearsay-II speech-understanding system: integrating knowledge to resolve uncertainty. *Computing Surveys*, 12(2):213–253, 1980.

[57] G. Ernst and A. Newell. *GPS: A Case Study in Generality and Problem Solving.* Academic Press, New York, NY, 1969.

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

[59] O. Faugeras and K. Price. Semantic description of aerial images using stochastic labeling. In *Proceedings: ARPA Image Understanding Workshop*, pages 89–94, McLean, VA, 1980.

[60] Richard Fikes and Tom Kehler. The role of frame-based representation in reasoning. *Communications of the ACM*, 28(9):904–920, 1985.

[61] M. A. Fishler. On the representation of natual scenes. In Allen R. Hanson and Edward M. Riseman, editors, *Computer Vision Systems*, Academic Press Inc., New York, NY, 1978.

[62] James D. Foley and Andries Van Dam. *Fundamentals of Interactive Computer Graphics.* Adison-Wesley Publishing Co., Reading, MA, 1982.

[63] Kenneth D. Forbus. *Qualitative Process Theory.* Technical Report 664, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1982.

[64] Kenneth D. Forbus. Qualitative reasoning about physical processes. In *Proceedings: Seventh International Joint Conference on Artificial Intelligence*, Vancouver, British Columbia, Canada, 1981.

[65] W. Frei and C. C. Chen. Fast boundary detection: a generalization and a new algorithm. *IEEE Tranactions on Computers*, 26(2):988–998, October 1977.

[66] K. S. Fu. Information processing of remotely sensed agricultural data. *Proceedings IEEE*, 57:639–653, 1969.

[67] Y. Fukada. Spatial clustering procedures for image analysis. *Pattern Recognition*, 12:395–403, 1980.

[68] J. P. Gambotto and O. Monga. A parallel and hierarchical algorithm for region growing. In *Proceedings: IEEE-CVPR Conference*, pages 649–651, 1985.

[69] T. D. Garvey. *Perceptual Strategies for Purposive Vision.* Technical Report 151, SRI International, 333 Ravenswood Ave., Melno Park, CA 94025, September 1977.

[70] Frank Glazer. *Hierarchical Motion Detection.* Ph.D. Dissertation, Computer and Information Science Department, University of Massachusetts, Amherst, Massachusetts 01003, 1987. Also as Technical Report 87-02 Computer and Information Science Department, University of Massachusetts.

[71] Frank Glazer. *Multilevel Relaxation in Low Level Computer Vision.* Technical Report 82-30, Computer and Information Science Department, University of Massachusetts, Amherst, Massachusetts 01003, 1982.

[72] A. Goldberg and D. Robson. *Smalltalk–80, the Language and the Implementation.* Addison–Wesley, Reading, MA, 1983.

[73] L. Van Gool, P. Dewaele, and A. Oosterlinck. Texture analysis anno 1983. *Computer Vision, Graphics, and Image Processing*, 29(3):336–357, March 1985.

[74] W. Eric L. Grimson and Theo Pavlidis. Discontinuity detection for visual surface reconstruction. *Computer Vision, Graphics, and Image Processing*, 30(3):316–330, June 1985.

[75] Guang-You.Xu and King-Sun.Fu. Natural scene segmentation based on multiple threshold and textural measures. In *Seventh International Conference on Pattern Recognition*, pages 1111–1113, July 1984.

[76] A. Guzman. Decomposition of a visual scene into three-dimensional bodies. In *AFIPS Fall Joint Conferences*, pages 291–304, 1968.

[77] A. Hanson, E. Riseman, P. Nagin, and R. Kohler. Segmentation, evaluation,and natural scenes. In *Conference on Pattern Recognition and Image Processing*, IEEE Computer Society, Chicago, Illinois, August 1979.

[78] A. R. Hanson, E. M. Riseman, and P. A. Nagin. Authors's reply. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(2):249, March 1984.

[79] Allen R. Hanson and Edward M. Riseman, editors. *Computer Vision Systems.* Academic Press Inc., New York, NY, 1978.

[80] Allen R. Hanson and Edward M. Riseman. A methodology for the development of general knowledge–based vision systems. In A. Hanson and M. Arbib, editors, *Vision, Brain, and Cooperative Computation*, MIT Press,

Cambridge, MA, 1987. Also as Technical Report 86-27, Computer and Information Science Department, University of Massachusetts, Amherst MA 01003.

[81] Allen R. Hanson and Edward M. Riseman. Processing cones: a computational structure for image analysis. In S. Tanimoto and A. Klinger, editors, *Structured Computer Vision*, pages 101–131, Academic Press Inc., New York, NY, 1981.

[82] Allen R. Hanson and Edward M. Riseman. Segmentation of natural scenes. In Allen R. Hanson and Edward M. Riseman, editors, *Computer Vision Systems*, pages 129–163, Academic Press Inc., New York, NY, 1978.

[83] Allen R. Hanson and Edward M. Riseman. Visions image understanding system - 1986. In Christopher M. Brown, editor, *Advances in Computer Vision*, Erlbaum Assoc., 1988. In Preparation.

[84] Allen R. Hanson, Edward M. Riseman, and Frank C. Glazer. *Edge Relxation and Boundary Continuity.* Technical Report 80-11, Computer and Information Science Department, University of Massachusetts, Amherst, Massachusetts 01003, 1980.

[85] Allen R. Hanson, Edward M. Riseman, and Paul R. Nagin. *Region-growing in Textured Outdoor Scenes.* Technical Report 75C-3, Computer and Information Science Department, University of Massachusetts, Amherst, Massachusetts 01003, 1982.

[86] R. M. Haralick. Zero-crossings of second directional derivative edge operator. In *Proceedings of the Society of Photo-Optical Instrumentation Engineers Technical Symposium East*, Arlington, VA, May 1982.

[87] R. M. Haralick and I. Dinstein. A spatial clustering procedure for multi-image data. *IEEE Transactions on Circuits and Systems*, CAS-22:440–450, 1975.

[88] R. M. Haralick and L. Watson. A facet model for image data. *Computer Graphics and Image Processing*, 15:113–129, 1981.

[89] Robert Haralick, K. Shanmugam, and Its'hak Dinstein. Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-3(6):610–631, November 1973.

[90] Robert M. Haralick. Digital step edges from zero crossing of second directional derivatives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(1):58–68, January 1984.

[91] Robert M. Haralick. *Statistical and Structural Approaches to Texture.* Technical Report, Department of Electrical Engineering and Computer Science, University of Kansas, Lawrence, Kansas, April 1976.

[92] Robert M. Haralick and Linda G. Shapiro. Image segmentation techniques. *Computer Vision, Graphics, and Image Processing*, 29(1):100–132, January 1985.

[93] Ralph Hartley. A Gaussian weighted multiresolution edge detector. *Computer Vision, Graphics, and Image Processing*, 30(1):70–83, 1985.

[94] G. G. Hendrix. Encoding knowledge in partitioned networks. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-3(6):610–631, November 1973.

[95] M. H. Heuckel. A local visual operator which recognizes edges and lines. *Journal of the ACM*, 20(4), October 1973.

[96] Ellen C. Hildreth. Edge detection in mand and machine. *Robotics Age*, 8–14, September 1981.

[97] Tsai-Hong Hong and Azriel Rosenfeld. *Unforced Image Partitioning by Weighted Pyramid Linking.* Technical Report TR-1137, Computer Vision Laboratory. University of Maryland, College Park, Maryland 20742, 1982.

[98] S. L. Horowitz and T. Pavlidis. Picture segmentation by a direct split-and-merge procedure. *Journal of the ACM*, 17:368–388, 1976.

[99] P. V. C. Hough. Method and means for recognizing complex patterns. 1962. U.S. Patent 3,069,654.

[100] D. H. Hubel and T. N. Wiesel. Receptive fields and functional architectures in two non–striate areas (18 and 19) of the cat. *J. Neurohysiol*, 28:228–289, 1965.

[101] D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interraction and functional architecture in the cat visual cortex. *J. Physiol*, 160:106–154, 1962.

[102] M. Hueckel. A local visual operator which recognizes edges and lines. *Journal of the ACM*, 20(4):634–637, October 1973.

[103] Andres Huertas and Gerard Medioni. Edge detection with subpixel precision. In *Proceedings: IEEE–CVPR Conference*, pages 633–636, 1985.

[104] D. A. Huffman. Impossible objects as nonsence sentences. *Machine Intelligence 6*, 1971.

[105] Vincent Shang-Shouq Hwang, Larry S. Davis, and Takashi Matsuyama. *Hypothesis Integration in Image Understanding Systems.* Technical Report TR-1137, Center for Automation Research, University of Maryland, College Park, Maryland 20742, June 1985.

[106] Vincent Shang-Shouq Hwang, Larry S. Davis, and Takashi Matsuyama. Hypothesis integration in image understanding systems. *Computer Vision, Graphics, and Image Processing*, 36(2):321–371, November 1986.

[107] A. Kay and A. Goldberg. Personal dynamic media. *Computer*, 9(10):31–41, 1977.

[108] J. R. Kender. Shape from texture: a brief overview and a new aggregation transform. In *Proceedings: DARPA Image Understanding Workshop*, pages 79–84, November 1978.

[109] John R. Kender. *Saturation, Hue, and Normalized Color: Calculation, Digitization Effects, and Use.* Technical Report, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA 15213, November 1976.

[110] John R. Kender. *Shape From Texture.* Ph.D. Dissertation, Department of Computer Science. Carnegie-Mellon University, November 1980.

[111] R. A. Kirsch. Computer determination of the constituent structure of biological images. *Computers and Biomedical Research*, 4(3):315–328, June 1971.

[112] J. Kittler, J. Illingworth, J. Foglein, and K. Paler. An automatic thresholding algorithm and its performance. In *Seventh International Conference on Pattern Recognition*, pages 287–289, July 1984.

[113] R. R. Kohler and A. R. Hanson. The visions image operating system. In *Proceedings: Sixth International Conference on Pattern Recognition*, pages 71–74, Munich, October 1982.

[114] Ralf Kohler. A segmentation system based on thresholding. *Computer Graphics and Image Processing*, 15:319–338, 1981.

[115] Ralf R. Kohler. *Integrating Non_semantic Knowledge into Image Segmentation Processes.* Ph.D. Dissertation, Computer and Information Science Department, University of Massachusetts, Amherst, Massachusetts 01003, 1984. Also as Technical Report 84-04, Computer and Information Science Department, University of Massachusetts.

[116] S. Kuffler. Discharge patterns and functional organization of mammalian retina. *J. Neurophysiol*, 16:37–68, 1953.

[117] K. A. Lanz, C. M. Brown, and D. H. Ballard. Model driven vision using procedural description: motivation and application to photointerpretation annd medical diagnosis. In *Proceedings 22nd International Symposium, Society of Photo-Optical Engineers*, San Diego, CA, August 1978.

[118] Kenneth Laws. *Textured Image Segmentation.* Ph.D. Dissertation, Image Processing Institute, University of Southern California, University Park, Los Angeles, CA 90007, January 1980.

[119] J. Y. Lettvin, H. Maturana, W. S. McCulloch, and W. H. Pitts. What the frog's eye tells the frog's brain. *Proc. IRE*, 47:1940–1951, 1959.

[120] M. Levine. A knowledge–based computer vision system. In Allen R. Hanson and Edward M. Riseman, editors, *Computer Vision Systems*, Academic Press Inc., New York, NY, 1978.

[121] M. D. Levine and A. F. Nazif. Dynamic measurement of computed image segmentations. *IEEE Transactions on Pattern Analysis and Machine Intellegence*, PAMI-7(2):155–164, 1985.

[122] Martin D. Levine and Samir I. Shaheen. A modular computer vision system for picture segmentation and interpretation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-3(5):540–554, September 1981.

[123] John D. Lowrance. *Dependency-Graph Models of Evidential Support.* Ph.D. Dissertation, Computer and Information Science Department, University of Massachusetts, Amherst, Massachusetts 01003, September 1982. Also as Technical Report 82-26, Computer and Information Science Department, University of Massachusetts.

[124] John D. Lowrance. *GRASPER 1.0 Reference Manual.* Technical Report 78-20, Computer and Information Science Department, University of Massachusetts, Amherst, Massachusetts 01003, December 1978.

[125] D. Marr and E. Hildreth. Theory of edge detection. *Proc. Royal Society of London*, B.207:187–217, 1980.

[126] David Marr. *Representing Visual Information.* Technical Report 415, Massachusetts Institute of Technology Artificial Intelligence Laboratory, Cambridge, Massachusetts, May 1977.

[127] David Marr. *Vision.* W. H. Freeman, San Francisco, CA, 1982.

[128] David Marr and H. K. Nisihara. Representation and recognition of the spatial organization of three dimensional shapes. *Proceedings of the Royal Society*, 200:269–294, 1977.

[129] Takashi Matsuyama. Knowledge organization and control structure in image understanding. In *Seventh International Conference on Pattern Recognition*, pages 1118–1127, July 1984.

[130] D. L. Milgram and M. Herman. Clustering edge values for threshold selection. *Computer Graphics and Image Processing*, 9:82–88, 1979.

[131] Marvin Minsky. *A Framework for Representing Knowledge.* Technical Report 306, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, June 1974.

[132] Johannes G. Moik. *Digital Signal Processing of Remotely Sensed Images.* NASA Scientific and Technical Information Branch, Washington D.C., 1980.

[133] M. Nagao and T. Matsuyama. Edge preserving smoothing. In *Proceedings: Fourth International Joint Conference on Pattern Recognition*, pages 518–520, 1978.

[134] M. Nagao and T. Matsuyama. *A Structural Analysis of Complex Aerial Photographs.* Plenum, New York, NY, 1980.

[135] M. Nagao, T. Matsuyama, and Y. Ikeda. Region extraction and shape analysis of aerial photographs. In *Proceedings: Fourth International Joint Conference on Pattern Recognition*, page 620, 1978.

[136] P. A. Nagin, A. R. Hanson, and E. M. Riseman. Studies in global and local histogram–guided relaxation algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-3:263–277, May 1982.

[137] Paul A. Nagin. *Studies in Image Segmentation Algorithms Based on Histogram Clustering and Relaxation.* Ph.D. Dissertation, Computer and Information Science Department, University of Massachusetts, Amherst, Massachusetts 01003, 1979. Also as Technical Report 79-15, Computer and Information Science Department, University of Massachusetts.

[138] Paul A. Nagin, Allen R. Hanson, and Edward M. Riseman. *Region Extraction and Description Through Planning.* Technical Report 77-8, Computer and Information Science Department, University of Massachusetts, Amherst Massachusetts 01003, May 1977.

[139] K. A. Narayanan, Dianne P. O'Leary, and Azriel Rosenfeld. *Multi-Resolution Relaxation.* Technical Report TR-1070, Computer Vision Laboratory University of Maryland, College Park, Maryland 20742, July 1981.

[140] Ahmed M. Nazif. *A Rule-Based Expert System for Image Segmentation.* Ph.D. Dissertation, Electrical Engineering Department, McGill University, Montreal, Canada, March 1983.

[141] Ahmed M. Nazif and Martin D. Levine. *Low Level Image Segmenation: An Expert System.* Technical Report TR-83-4, Computer Vision and Robotics Laboratory, McGill University, April 1983.

[142] U. Neisser. *Cognitive Psychology.* Appleton-Century-Crofts, New York, NY, 1967.

[143] R. Nevatia and K. R. Babu. Linear feature extraction and description. In *Proceedings: Seventh International Joint Conference on Artificial Intelligence*, pages 639–641, 1981.

[144] Charles F. Neveu, Charles R. Dyer, and Roland T. Chin. Object recognition using hough pyramids. In *Proceedings: IEEE–CVPR Conference*, pages 328–333, 1985.

[145] L. O'Gorman and A. C. Sanderson. The converging squares algorithm: a efficient method for locating peaks in multidimensions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(3):280–287, March 1984.

[146] R. Ohlander, K. Price, and D. R. Reddy. Picture segmentation using a recursive region splitting method. *Computer Graphics and Image Processing*, 8:313–333, 1978.

[147] Ronald B. Ohlander. *Analysis of Natural Scenes.* Ph.D. Dissertation, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA 15213, April 1975.

[148] Yu-ichi Ohta. *Knowledge-based Interpretation of Outdoor Natural Color Scenes.* Pitman Advanced Publishing Program, Boston, MA, 1985.

[149] Yu-ichi Ohta. *A Region–Oriented Image–Analysis System by Computer.* Ph.D. Dissertation, Department of Computer Science, Kyoto University, 1980.

[150] J. O'Rourke and K. R. Sloan Jr. Dynamic quantization: two adaptive data structures for multidimensional squares. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(3):266–279, May 1984.

[151] K. J. Overton and T. E. Weymouth. A noise reducing preprocessing algorithm. In *Proceedings: Pattern Recognition and Image Processing*, pages 498–507, Chicago, IL, 1979.

[152] Cesare C. Parma, Allen R. Hanson, and Edward M. Riseman. *Experiments in Schema-Driven Interpretation of A Natural Scene.* Technical Report 80-10, Computer and Information Science Department, University of Massachusetts, Amherst, Massachusetts 01003, April 1980.

[153] S. Peleg, J. Naor, R. Hartly, and D. Avnir. Multiple resolution texture analysis and classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(4):518–522, July 1984.

[154] Matti Pietikainen and Azriel Rosenfeld. *Edge-Based Texture Measures.* Technical Report TR-1050, Computer Science, University of Maryland, College Park, Maryland 20742, May 1981.

[155] Matti Pietikainen and Azriel Rosenfeld. *Gray Level Pyramid Linking as an Aid in Texture Analysis.* Technical Report TR-1061, Computer Science, University of Maryland, College Park, Maryland 20742, July 1981.

[156] K. K. Pingle and J. M. Tenenbaum. An accomodating edge follower. In *Proceedings: Second International Joint Conference on Artificial Intelligence*, pages 1–7, September 1971.

[157] Ting–Chuen Pong, Linda G. Shapiro, Layne T. Watson, and Robert M. Haralick. Experiments in segmentation using a facet model region grower. *Computer Vision, Graphics, and Image Processing*, 25(1):1–23, January 1984.

[158] William K. Pratt. *Digital Image Processing.* Wiley, New York, NY, 1978.

[159] J. M. S. Prewitt. Object enhancement and extraction. In B. S. Lipkin and A. Rosenfeld, editors, *Picture Processing and Psychophysics*, Academic Press, New York, NY, 1970.

[160] K. Price. Image segmenation: a comment on studies in global and local histogram-guided relaxation algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(2):247–248, March 1984.

[161] K. E. Price and R. Reddy. Matching segments of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI:1:110–116, 1979.

[162] G. Reynolds, D. Strahman, and N. Lehrer. Converting feature values to evidence. In *Proceedings: DARPA Image Understanding Workshop*, Miami Beach, FL, 1985.

[163] George Reynolds and J. Ross Beveridge. Searching for geometric structure in images of natural scenes. In *Proceedings: DARPA Image Understanding Workshop*, pages 257–271, Morgan Kaufman, February 1987.

[164] George Reynolds, Nancy Irwin, Allen Hanson, and Edward Riseman. Hierarchical knowledge-directed object extraction using a combined region and line representation. In *Proceedings Of The IEEE Workshop On Computer Vision Representation And Control*, pages 238–247, 1984.

[165] Edward M. Riseman and Michael A. Arbib. Computational techniques in the visual segmentation of static scenes. *Computer Graphics and Image Processing*, 6:221–276, 1977.

[166] Edward M. Riseman and Allen R. Hanson. *A Methodology for the Development of General Knowledge-Based Vision Systems.* Technical Report 86-27, Computer and Information Science Department, University of Massachusetts, Amherst, Massachusetts 01003, 1986.

[167] R.L.Kashyao and Kie-Bum Eom. Texture boundary detection using long correlation model. In *1985 International Geoscience and Remote Sensing Symposium*, pages 255–259, IEEE Geoscience and Remote Sensing Society, 1985.

[168] L. G. Roberts. Machine perception of three-dimensional solids. In J. P. Tippett, editor, *Optical and Electro-optical Information Processing*, MIT Press, Cambridge, MA, 1965.

[169] A. Rosenfeld, R. A. Hummel, and S. W. Zucker. *Scene Labelling by Relaxation Operators.* Technical Report TR-379, Computer Vision Laboratory University of Maryland, College Park, Maryland 20742, 1975.

[170] A. Rosenfeld and A. Kak. *Digital Picture Processing.* Academic Press, New York, NY, 1976.

[171] David A. Rosenthal. *An Inquiry Driven Computer Vision System Based On Visual and Conceptual Hierarchies.* Ph.D. Dissertation, Moore School of Electrical Engineering, University of Pennsylvania, Philadelphia, PA 19104, 1978.

[172] H. Samet. Region representation: quadtrees from boundary codes. *Communications of the ACM*, 23(3):163–170, March 1980.

[173] Alireza Sarabi and J. K. Aggarwal. Segmentation of chromatic images. *Pattern Recognition*, 13(6):417–427, 1981.

[174] P. G. Selfridge. *Reasoning About Success and Failure in Aerial Image Understanding.* Ph.D. Dissertation, Computer Science Department, University of Rochester, 1982. Also as Technical Report 101, Computer Science Department, University of Rochester, Rochester NY.

[175] Steven Shafer and Takeo Kanade. Recursive region segmentation by analysis of histograms. In *Proceedings: Sixth International Joint Conference on Pattern Recognition*, pages 1166–1171, Munich, Germany, October 1982.

[176] Bert Shaw. *Some Remarks on the Use of Color for Machine Vision.* Technical Report 83-31, Computer and Information Science Department, University of Massachusetts, Amherst, Massachusetts 01003, 1983.

[177] Y. Shirai. A context sensitive line finder for recognition of polyhedra. *Artificial Intelligence*, 4:95–119, 1973.

[178] Y. Shirai. Recognition of man–made objects using edge cues. In Allen R. Hanson and Edward M. Riseman, editors, *Computer Vision Systems*, Academic Press Inc., New York, NY, 1978.

[179] Michael Shneier. *Two Hierarchical Linear Feature Representations: Edge Pyramids and Edge Quadtrees.* Technical Report TR-961, Computer Vision Laboratory University of Maryland, College Park, Maryland 20742, 1980.

[180] E. H. Shortliffe. *Computer Based Medical Consultations: MYCIN.* North-Holland, New York, NY, 1976.

[181] R. Southwick. *FEATSYS - An Intermediate–Level Representation of Image Feature Data.* Master's thesis, Computer and Information Science Department, University of Massachusetts, Amherst, MA 01003, 1986.

[182] Steven L. Tanimoto. Regular hierarchical image and processing structures in machine vision. In Allen R. Hanson and Edward M. Riseman, editors, *Computer Vision Systems*, pages 165–174, Academic Press Inc., New York, NY, 1978.

[183] J. M. Tenenbaum and H. G. Barrow. *Experiments in Interpretation-Guided Segmentation.* Technical Report 123, SRI International, 333 Ravenswood Ave., Melno Park, CA 94025, March 1976.

[184] William B. Thompson and Albert Yonas. What should be computed in low level vision. In *Proceedings of The First Annual National Conference on Artificial Intelligence*, pages 7–10, August 1980.

[185] E. Triendl and T. Henderson. A model for texture edges. In *Proceedings: Fifth International Conference on Pattern Recognition*, pages 1100–1102, December 1980.

[186] Mohan M. Trivedi, Charles A. Harlow, Richard W. Conners, and Semoon Goh. Object detection based on gray level cooccurrence. *Computer Vision, Graphics, and Image Processing*, 28(2):199–219, November 1984.

[187] J. K. Tsotsos. Knowledge of the visual process: content, form, and use. In *Proceedings: Sixth International Joint Conference on Pattern Recognition*, pages 654–669, Munich, Germany, October 1982.

[188] Lewis W. Tucker. Control strategy for an expert vision system using quadtree refinement. In *Proceedings of the Workshop on Computer Vision: Representation and Control*, pages 214–218, IEEE, April 1984.

[189] S. A. Underwood and J. K. Aggarwal. Interactive computer analysis of aerial color infrared photographs. *Computer Vision, Graphics, and Image Processing*, 6(1):1–24, Feburary 1977.

[190] D. Waltz. Generating semantic descriptions from drawings of scenes with shadows. In P. Winston, editor, *The Psychology of Computer Vision*, pages 19–92, McGraw-Hill, New York, NY, 1972.

[191] Charles C. Weems, Steven P. Levitan, Allen R. Hanson, and Edward M. Riseman. The image understanding architecture. In *Proceedings: DARPA Image Understanding Workshop*, pages 483–498, Morgan Kaufman, February 1987. Also as Technical Report 87-76, Computer and Information Science Department, University of Massachusetts, Amherst MA 01003.

[192] Richard Weiss and Michael Boldt. Geometric grouping applied to straight lines. In *Proceedings: IEEE-CVPR Conference*, pages 489–495, June 1986.

[193] Diederich Wermser. Unsupervised segmentation by use of a texture gradient. In *Seventh International Conference on Pattern Recognition*, pages 1114–1116, July 1984.

[194] L. Wesley and A. Hanson. Evidential knowledge-based computer vision. *Optical Engineering*, 25(3):363–379, March 1988.

[195] Leonard Wesley. *The Application of an Evidential Based Technology to a High-Level Knowledge-Based Image Interpretation System.* Ph.D. Dissertation, Computer and Information Science Department, University of Massachusetts, Amherst, Massachusetts 01003, 1988.

[196] T. E. Weymouth, J. S. Griffith, A. R. Hanson, and E. M. Riseman. Rule based strategies for image interpretation. In *Proceedings: DARPA Image Understanding Workshop*, pages 193–202, Arlington, VA, June 1983.

[197] Terry E. Weymouth. *Using Object Descriptions in a Schema Network for Machine Vision.* Ph.D. Dissertation, Computer and Information Science Department, University of Massachusetts, Amherst, Massachusetts 01003, 1986. Also as Technical Report 86-24 Computer and Information Science Department, University of Massachusetts.

[198] D. L. Williams and M. L. Stouffer. Monitoring gypsy moth defoliation via Landsat image differencing. In *Symposium on Remote Sensing for Vegetation Damage Assesment*, pages 221–229, 1978.

[199] Lance R. Williams and P. Anandan. *A Coarse-to-Fine Control Strategy for Stereo and Motion on a Mesh-Connected Computer.* Technical Report 86-19, Computer and Information Science Department, University of Massachusetts, Amherst, Massachusetts 01003, December 1986.

[200] Thomas D. Williams. *Computer Interpretation of a Dynamic Image from a Moving Vehicle.* Ph.D. Dissertation, Computer and Information Science Department, University of Massachusetts, Amherst, Massachusetts 01003, May 1981.

[201] Andrew P. Witkin. Recovering surface shape and orientation from texture. *Artificial Intelligence*, 17:17–45, 1981.

[202] William A. Woods. What's important about knowledge representation. *Computer*, 16(10):22–29, October 1983.

[203] Y. Yakimovsky. *Scene Analysis Using a Semantic Base for Region Growing.* Ph.D. Dissertation, Stanford University, Palo Alto, CA, 1973.

[204] Lotfi A. Zadeh. Commonsense knowledge representation based on fuzzy logic. *Computer*, 16(10):61–65, October 1983.

[205] A.L. Zobrist and G. Nagy. Pictorial information processing of landsat data for geographic analysis. *Computer*, 14(11):34–42, November 1981.

[206] S. W. Zucker. Relaxation labeling and the reduction of local ambiguity. In C. H. Chen, editor, *Pattern Recognition and Artificial Intelligence*, Academic Press, New York, NY, 1977.

[207] S. W. Zucker, A. Rosenfeld, and L. S. Davis. Picture segmentation by texture discrimination. *IEEE Tranactions on Computing*, C-24(12):1228–1233, 1975.