REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED FINAL/30 Sep 93 TO 29 Dec 94

4. TITLE AND SUBTITLE
MULTISTRATEEGY LEARNING FOR IMAGE UNDERSTANDING

5. FUNDING NUMBERS
A414/02
F49620-93-1-0624

6. AUTHOR(S)
BIR BHANU

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
UNIVERSITY OF CALIFORNIA
COLLEGE OF ENGINEERING
RIVERSIDE, CA 92521-0425

8. PERFORMING ORGANIZATION REPORT NUMBER
AFOSR-TR- 95 - 0298

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFOSR/NM
110 DUNCAN AVE, SUTE B115
BOLLING AFB DC 20332-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
F49620-93-1-0624

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE: DISTRIBUTION IS UNLIMITED

12b. DISTRIBUTION CODE

Current Image Understanding (IU) algorithms and systems lack the flexibility and robustness to successfully handle complex real-world situations. Robust 3-D object recognition, in real-world applications operating under changing environmental conditions, remains one of the important but elusive goals of IU research. We believe that an innovative combination of IU and Machine Learning (ML) techniques will lead to the advancement of the IU filed in general. IU itself has come to a certain state of maturity, in that we have today a good understanding of the essential components, their functionality, and the architectural issues involve. IU processes are commonly separated into three hierarchical layers, called the low, intermediate, and high level. At each of these levels. ML techniques can be employed selectively to improve the overall recognition performance. By introducing adaptation of task parameters; maintenance of internal representations and hypotheses pertaining to the observed reality: and learning new concepts and recognition strategies. The incorporation of learning into IU algorithms and systems will results in adaptation and robustness capability since learning provides automatic knowledge acquisition and continuous improvement of recognition system performance.

14. SUBJECT TERMS

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR(SAME AS REPORT)

Multistrategy Learning for Image Understanding

Technical Report

September 30 1993 to December 29, 1994
Grant Number F49620-93-1-0624

February 15, 1995

Prepared for the
Air Force Office of Scientific Research
Sponsor Code 2416
and the
Advanced Research Projects Agency

by

Bir Bhanu
Principal Investigator

University of California
College of Engineering
Riverside, CA 92521-0425

Contributors:

Bir Bhanu
Dongsung Kim
Wilhelm Burger
Subhodev Das
Rabi Dutta
Jing Peng
Songnain Rong
Xing Wu

Contents

List of Figures

List of Tables

Chapter 1

Summary

Current Image Understanding (IU) algorithms and systems lack the flexibility and robustness to successfully handle complex real-world situations. Robust 3-D object recognition, in real-world applications operating under changing environmental conditions, remains one of the important but elusive goals of IU research. We believe that an innovative combination of IU and Machine Learning (ML) techniques will lead to the advancement of the IU field in general. IU itself has come to a certain state of maturity, in that we have today a good understanding of the essential components, their functionality, and the architectural issues involved. IU processes are commonly separated into three hierarchical layers, called the low, intermediate, and high level. At each of these levels, ML techniques can be employed selectively to improve the overall recognition performance: by introducing adaptation of task parameters; maintenance of internal representations and hypotheses pertaining to the observed reality; and learning new concepts and recognition strategies. The incorporation of learning into IU algorithms and systems will results in adaptation and robustness capability since learning provides automatic knowledge acquisition and continuous improvement of recognition system performance.

Current computational learning taxonomy identifies five major ML paradigms, based upon representation schemes and learning methods employed:

- *Inductive Learning* uses experience to generate a conjecture and uses further experience to confirm or refute it. It is useful to synthesize new knowledge.

- *Analytical Learning* (e.g., Explanation-based Learning) uses observations to generate conjectures and attempts to confirm these as logical consequences of the existing knowledge. It is useful to improve existing knowledge.

- *Case-Based Reasoning* uses memory of relevant "past" cases to interpret or solve a new problem case. It is useful when cases similar to its current situation are encountered. It can help to avoid past mistakes.
- *Genetic Algorithms* are a family of adaptive search methods that are modeled after genetic evolution process with the advantage that the search process is independent of the problem domain.
- *Connectionist Learning* is biologically inspired and uses parallel distributed networks of computational nodes with the advantage that the network adaptation is independent of the problem domain.

The general approach for selecting a computational learning technique for a given application involves the following steps [22]: (1) understand the task well enough to select appropriate functions for evaluating the performance of the learning system; (2) abstract and define a learning problem from the task problem; (3) select a particular computational learning paradigm for the abstracted learning problem. In practice, these three steps are not completely separable. We have found, however, that this general process is valuable for uncovering non-obvious — but compelling — applications of machine learning in the domain of object recognition.

1.1 Multistrategy Learning in Image Understanding

The application of ML to the IU domain is more demanding than most conventional learning applications in AI. It is caused by (a) the enormous amount of incoming data to be processed, and (b) the variety of processes and representations encountered in Image Understanding. Consequently, the selection and adaptation of existing learning techniques require careful attention. Due to the variety of tasks involved, we cannot expect a single learning technique to solve the entire learning problem in IU, but different learning strategies must be optimally combined to achieve the desired results.

The Multi strategy Learning-Based IU System (Figure 1.1) selectively applies machine learning techniques at multiple levels to achieve robust recognition performance.

The system uses Genetic Algorithms (GAs) to optimize multi-sensor image segmentation at the low level. At the intermediate level, Explanation-Based Learning (EBL) is employed to learn new visual concepts for improving indexing and matching. At the high-level, Case-Based Reasoning (CBR) is used to dynamically adapt recognition strategies, and acquiring and maintaining information about the environment. At each level, appropriate evaluation criteria are employed to monitor the performance and self-improvement of the system.

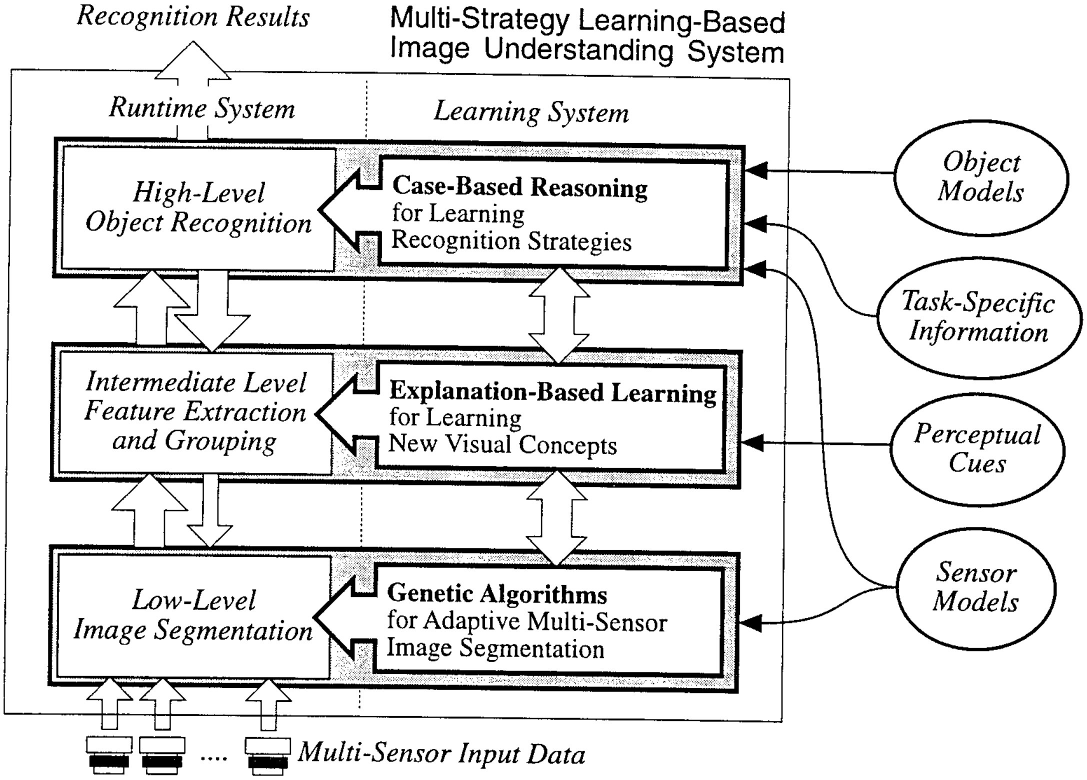

Recognition Results
Multi-Strategy Learning-Based Image Understanding System
Runtime System
Learning System
High-Level Object Recognition
Case-Based Reasoning for Learning Recognition Strategies
Intermediate Level Feature Extraction and Grouping
Explanation-Based Learning for Learning New Visual Concepts
Low-Level Image Segmentation
Genetic Algorithms for Adaptive Multi-Sensor Image Segmentation
Object Models
Task-Specific Information
Perceptual Cues
Sensor Models
.... Multi-Sensor Input Data

Figure 1.1: Multistrategy Learning-Based IU System

1.2 Learning at the Low Level: Adaptive Multi-Sensor Image Segmentation

Image segmentation is an extremely important and difficult low-level task. All subsequent interpretation tasks including object detection, feature extraction, object recognition, and classification rely heavily on the quality of the segmentation process. The difficulty arises when the segmentation performance needs to be adapted to the changes in image quality that is affected by variations in environmental conditions, imaging devices, time of day, etc. Despite the large number of segmentation techniques presently available [39, 50] no general methods have been found that perform adequately across a diverse set of imagery, i.e., no segmentation algorithm can automatically generate an "ideal" segmentation result in one pass (or in an open loop manner) over a range of scenarios encountered in practical unstructured applications. Any technique, no matter how "sophisticated" it may be, will eventually yield poor performance if it cannot adapt to the variations in unstructured scenes.

Genetic Algorithms (GAs) are efficient in locating an approximate global maximum in a search space and therefore show great promise in solving the parameter selection problem encountered in the image segmentation task. They use simple recombinations of existing high quality search points together with a method of measuring current performance. Our initial research has demonstrated that adaptive image segmentation can provide over 30% improvement in performance over non-adaptive techniques[20].

The objectives of our genetics-based adaptive image segmentation system working under varying scenarios and/or environmental conditions are (a) learning the optimal parameter settings for adaptive image segmentation, (b) learning the optimal selection of image segmentation algorithms, for multi-scenarios, and (c) learning the optimal sensor combinations and cross-sensor validation of segmentation results.

1.3 Learning at the Intermediate Level: Learning Composite Visual Concepts

Model-based object recognition methods require image data to be matched with models in the system database. Typically, the image data consists of unordered sets of simple geometric primitives like lines, arcs, and corners. It is well known that the computational complexity of the matching process is exponential in the number of image features for a given object model. Grouping has been shown to be an effective means for reducing the search complexity in structural matching [63]. However, most current grouping techniques use only perceptually motivated, low-order geometrical relationships, (such as collinearity, cotermination, parallelism, proximity, etc.), but no object model information, to assemble simple features of the same type. As a result, the full potential of grouping for solving the indexing problem has not been realized.

The key features of our approach at the intermediate level are:

1. The use of a two-stage grouping strategy that combines (a) domain-independent perceptual grouping and (b) model-based grouping with a database of high-order structural arrangements (intermediate visual concepts).

2. The use of Explanation-Based Learning to automatically infer the most useful intermediate visual concepts from real examples.

3. The introduction of multiple types of primitive features ("polymorphic" features) to participate in feature groupings instead of a single feature type, which will lead to increased robustness (by providing redundancy) and indexing power.

1.4 Learning at the High Level: Learning Recognition Strategies in Dynamic Environments

Automatic acquisition of recognition strategies in dynamic situations has been a bottleneck in the development of automated IU systems applied to real-world problems. The problem occurs while matching a stored object model to an input instance of that model and is attributed to the initially unknown pose of object and the varying environmental conditions.

During the process of image/scene understanding a human relies heavily on the memory of past cases and experience. For this purpose, we use the Case-Based Reasoning (CBR) paradigm in which "past" experiences are stored in memory as cases and are used to solve a new problem case. Similar cases can be combined to create problem solving shortcuts or to anticipate problems in new situations. The set of cases is prioritized and a strategy for the current problem is generated and executed. Various combinations of cases are created until a successful solution is reached.

The key features of our high-level approach are:

1. Cases are represented in a hierarchical manner and include task related knowledge, context, and recognition strategies. Explanation-Based Learning (EBL) is also used for generalizing and refining the individual cases before they are stored in the case memory.
2. Visual concepts instantiated at the intermediate level are used for indexing and matching, in conjunction with the evaluation of recognition performance.
3. Similarities and differences are used to match new situations to existing cases.

1.5 Interaction and Feedback between Multiple Levels

Current computer vision systems for model-based object recognition are open-loop systems that typically use image segmentation followed by object recognition algorithms. As a result of the open-loop nature, these systems are not robust for most real-world problems. With the goal of achieving robust recognition performance by using closed-loop systems, the question is how we can control feedback in a systematic manner from high-level recognition to low-level image segmentation. This has been a long-standing problem in the field of computer vision. Our approach for closed-loop model-based object recognition determines its criteria for image segmentation or feature extraction by using the recognition algorithm as part of the evaluation function. The confidence level of the matching algorithm serves as a reinforcement signal to generate the new values for segmentation or feature extraction parameters. This results in the performance improvement of the recognition system and generation of recognition strategies automatically.

1.6 Other Research in This Area

The incorporation of machine learning techniques into IU has been limited to isolated problems but there have been no approaches to integrate learning at multiple levels [22].

Learning in Image Understanding — Recent learning-based approaches to model-based object recognition have primarily focused on the connectionist framework or have emphasized object model acquisition. However, in all of these past approaches, the problem of object recognition has been treated as self-contained, independent of any lower-level feature extraction problems that may be encountered in reality. In contrast, our proposed approach treats the object recognition problem as a multi-level (low, intermediate, and high level) vision task.

Learning in Low-Level Vision — Genetic algorithms have been used for learning algorithm parameters for adaptive segmentation of color and textured images as well as the adaptive extraction of parametric image curves. Approaches using artificial neural nets (ANN) have been developed for adaptive image segmentation using perceptually motivated features. There has been some work in automatic sensor modeling, however, the problem of sensor, algorithm/parameter selection has not been addressed.

Learning in Intermediate-Level Vision — Structural feature detection is usually based on a fixed set of visual primitives for which efficient detection algorithms are available. The incorporation of features of varying complexity has been addressed using only non-adaptive, domain-independent grouping criteria. The problem of automatically forming intermediate-level perceptual shape concepts has found considerable attention in the psychological field recently, and interesting computational theories have emerged which, however, have not been implemented in IU architectures.

Learning in High-Level Vision — Learning by analogy was the first approach to incorporate a learning capability into a computer vision program, followed by learning class descriptions from examples. An inductive learning system was developed that creates production rules for recognizing isolated 2-D objects. ANN-based schemes have been used for recognition based on multivariate approximation theory. A multistrategy learning technique that incorporates Explanation-Based Learning and Structured Conceptual Clustering techniques has been used to automatically acquire and refine 2-D aircraft models.

Evaluation of Vision Algorithms — Very little work has been done to date on systematic evaluation of vision algorithm and system performance. Most current evaluation methods are selected ad hoc, performed off-line, and not integrated into IU systems architectures. Statistical approaches, such as the surface response method have been used for algorithm evaluation in conjunction with parameter selection. No practical IU systems exist today that provide performance evaluation of the overall system by making use of the inherent performance evaluation of individual learning components.

1.7 Accomplishments

During the reporting period, we have made the following achievements:

(a) *Genetic Algorithm for Adaptive Image Segmentation* (Chapter 2): We find that our genetic learning-based adaptive image segmentation approach scales with respect to the number of parameters and the size of the search space. Genetic learning combined with a hill-climbing technique is able to adaptively select good segmentation parameters and generate the best result using the least number of segmentations. By designing the experiments to evaluate the scalability of our approach, we find that when the size of the search space for four Phoenix parameters is 1 million, we search about 0.5% of the search space. This needs to be compared to the situation when we adapt two Phoenix parameters, the size of search space is 1024 and 2.4% of the search space is examined to find the global maximum.

(b) *Learnable Structural Models for Target Indexing* (Chapters 3 and 4): We have developed an approach to indexing that is based on utilization of *weak* structural models, *direct* table look-up, and *inexact* sequence matching. The weak structural models are defined as hidden Markov models (HMMs) which together with inexact analysis are appropriate for handling uncertainties and distortion in the imaging process; the table look up method utilizes invariant features similar to the existing approaches to indexing. The HMMs, the look-up table, and the database for sequence matching are all learned from training examples.

Using Gabor wavelet preprocessing we have developed techniques for quantization of local image measurements, learned HMM parameters from real examples and developed techniques for sequentialization of observations.

(c) *Reinforcement Learning for Closed-Loop Object Recognition* (Chapter 5 and 6): Using the *Phoenix* algorithm for the segmentation of color images, a clustering-based algorithm for the recognition of occluded 2-D objects and a *team of learning automata* algorithm, we show that in simple real scenes with varying environmental conditions and camera motion, effective low-level image analysis can be performed. We show the performance improvement of an IU system combined with learning over an IU system with no learning.

We have also developed a closed-loop object recognition system based on delayed reinforcement learning methods, where we learn segmentation and feature extraction parameters. In the future we plan to compare reinforcement learning algoithms based on team of learning automata and delayed reinforcement.

(d) *Context Reinforced Background Modeling* (Chapter 7): We have developed a technique that uses reinforcement learning to relate background models with the context for automatic target detection. Background models are represented by a novel self-organizing approach that uses both positive and negative examples to improve classification performance.

(e) *Case-Based Learning of Recognition Strategies* (Chapter 8): We have developed an approach to model-based object recognition under real-world conditions using Case-Based Reasoning (CBR) paradigm. This paradigm is analogical to human reasoning process which relies heavily on the memory of past cases and experience. Using CBR, successful recognition strategies are stored in memory as cases and are used to solve a new problem. Various combinations of cases are created until a successful solution is reached for the new situation.

(f) *Learning Composite Visual Concepts* (Chapter 9): We have specified the goals, prerequisites, and a preliminary formalism for "inventing" significant structural groupings from multi-class primitives. The approach is based on discovering groupings that have both a simple description and are distinctive for indexing into the model base, using a variant of explanation-based learning.

(g) *Consolidated Recognition and Motion Analysis* (Chapter 10): Using two sequences of outdoor and indoor color images, we show preliminary results for performance improvement in recognition by the interaction of color and dense range images obtained from motion analysis. The objects in the sequences are quite simple (traffic cones, cans, etc.).

Other accomplishments include the publication of a book on "Genetic Learning for Adaptive Image Segmentation," publication of a chapter in a book, submission of two papers to a special issue of Proceedings of the IEEE on "Signals and Symbols," and papers to IJCAI'95, ICCV'95 and IUW'94.

Chapter 2

Genetic Algorithm for Adaptive Image Segmentation

2.1 Introduction

Image segmentation is an old and difficult problem. It refers to the grouping of parts of an image that have "similar" image characteristics. All subsequent interpretation tasks including object detection, feature extraction, object recognition, and classification rely heavily on the quality of the segmentation process. The difficulty arises when the segmentation performance needs to be adapted to the changes in image quality. Image quality is affected by variations in environmental conditions, imaging devices, time of day, etc. Despite the large number of segmentation techniques presently available [39, 50], no general methods have been found that perform adequately across a diverse set of imagery, i.e., no segmentation algorithm can automatically generate an "ideal" segmentation result in one pass (or in an open loop manner) over a range of scenarios encountered in practical applications. Any technique, no matter how "sophisticated" it may be, will eventually yield poor performance if it cannot adapt to the variations in real-world scenes. The following are the key characteristics of the image segmentation problem:

- When presented with a new image, selecting the appropriate set of algorithm parameters is the key to effectively segmenting the image. Most segmentation techniques contain numerous **control parameters** which must be adjusted to obtain optimal performance, i.e., they are to be *learned.* The size of the parameter search space in these approaches can be prohibitively large, unless it is traversed in a highly efficient manner.

- The parameters within most segmentation algorithms typically interact in a complex, non-linear fashion, which makes it difficult or impossible to model the parameters' behavior in an algorithmic or rule-based fashion.

- The variations between images cause changes in the segmentation results, the objective function that represents segmentation quality varies from image to image. The search technique used to optimize the objective function must be able to adapt to these variations.

- The definition of the objective function itself can be a subject of debate because there are no universally accepted measures of image segmentation quality.

Hence, a need exists to apply an adaptive technique that can efficiently search the complex space of plausible parameter combinations and locate the values which yield optimal results. The approach should not be dependent on the particular application domain nor should it have to rely on detailed knowledge pertinent to the selected segmentation algorithm. Genetic algorithms (GA), which are designed to efficiently locate an approximate global maximum in a search space, have the attributes described above and show great promise in solving the parameter selection problem encountered in the image segmentation task.

The next section of this Chapter argues about the genetic algorithms as the appropriate optimization technique for the segmentation problem. Section 3 describes the adaptive image segmentation algorithm. We explain the choice of a particular segmentation algorithm as well as the manner in which segmentation quality is measured. Section 4 presents the experimental results on a sequence of outdoor images. We compare adaptive image segmentation results with other non-adaptive segmentation techniques. Section 5 presents the adaptive segmentation results when we scale the number of parameters in a scheme that uses genetic algorithms and hill climbing. Finally, Section 6 provides the conclusions of this Chapter.

2.2 Image Segmentation as an Optimization Problem

Fig. 2.1 provides an example of an objective function that is typical for the image segmentation process. The figure depicts an application in which only two segmentation parameters (*maxmin* and *absscore*) are being varied, and the corresponding segmentation quality obtained for any pair of algorithm parameters. Because the algorithm parameters interact in complex ways, the objective function is multimodal and presents problems for many commonly used optimization techniques. Further, since the surface is derived from an analysis

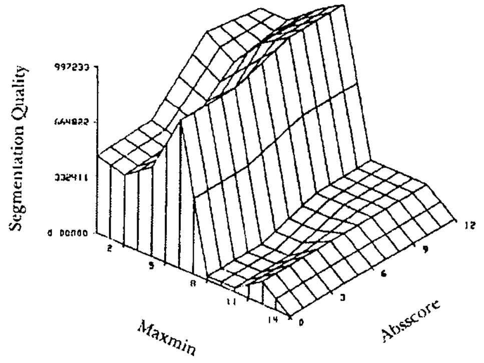


Figure 2.1: Segmentation quality surface.

of real-world imagery, it may be discontinuous, may contain significant amounts of noise, and cannot be described in closed form. The derivation of this surface will be described in Section 3, where we discuss the segmentation evaluation process.

The conclusion drawn from an analysis of various segmentation quality surfaces that we have examined is that we must utilize a highly effective search strategy which can withstand the breadth of performance requirements necessary for the image segmentation task.

2.2.1 Parameter Search Techniques

Various commonly used search techniques for functional optimization exist. The drawbacks to each of these methodologies are as follows:

- Exhaustive Techniques (Random walk, depth first, breadth first, enumerative) — Able to locate global maximum but computationally prohibitive because of the size of the search space.
- Calculus-Based Techniques (Gradient methods, solving systems of equations) — No closed form mathematical representation of the objective function is available. Discontinuities and multi-modal complexities are present in the objective function.
- Partial Knowledge Techniques (Hill climbing, beam search, best first, branch and bound, dynamic programming, A*) — Hill climbing is plagued by the foothill, plateau, and ridge problems. Beam, best first, and A* search techniques have no available

measure of goal distance. Branch and bound requires too many search points while dynamic programming suffers from the curse of dimensionality [110].

- Knowledge-Based Techniques (Production rule systems, heuristic methods) — These systems have a limited domain of rule applicability, tend to be brittle [53], and are usually difficult to formulate. Further, the visual knowledge required by these systems may not be representable in knowledge-based formats.

There are other search techniques such as genetic algorithms, simulated annealing and hybrid method [16]. To address the characteristic of image segmentation problem as discussed earlier, we have selected genetic algorithms for adaptive image segmentation.

2.2.2 Genetic algorithms for Image Segmentation

Genetic algorithms were pioneered at the University of Michigan by John Holland and his associates [30, 43, 52]. The term genetic algorithm is derived from the fact that its operations are loosely based on the mechanics of genetic adaptation in biological systems. Genetic algorithms can be briefly characterized by three main concepts: a Darwinian notion of fitness or strength which determines an individual's likelihood of affecting future generations through reproduction; a reproduction operation which produces new individuals by combining selected members of the existing population; and genetic operators which create new offspring based on the structure of their parents.

A genetic algorithm maintains a constant-sized population of candidate solutions, known as individuals. The initial seed population from which the genetic process begins can be chosen randomly or on the basis of heuristics, if available for a given application. At each iteration, known as a generation, each individual is evaluated and recombined with others on the basis of its overall quality or fitness. The expected number of times an individual is selected for recombination is proportional to its fitness relative to the rest of the population. Intuitively, the high strength individuals selected for reproduction can be viewed as providers of "building blocks" from which new, higher strength offspring can be constructed. An abstract procedure of a simple genetic algorithm is given below, where P(t) is a population of candidate solutions to a given problem at generation t.

```
t = 0;
initialize P(t);
evaluate P(t);
while not <termination condition>
        begin
                $t = t + 1;$
        reproduce P(t) from P(t-1);
        recombine P(t);
        evaluate P(t);
    end;
```

New individuals are created using two main genetic recombination operators known as crossover and mutation. Crossover operates by selecting a random location in the genetic string of the parents (crossover point) and concatenating the initial segment of one parent with the final segment of the second parent to create a new child. A second child is simultaneously generated using the remaining segments of the two parents. The string segments provided by each parent are the building blocks of the genetic algorithm. Mutation provides for occasional disturbances in the crossover operation by inverting one or more genetic elements during reproduction. This operation insures diversity in the genetic strings over long periods of time and prevents stagnation in the convergence of the optimization technique.

The individuals in the population are typically represented using a binary notation to promote efficiency and application independence of the genetic operations. Holland [52] provides evidence that a binary coding of the genetic information may be the optimal representation. Other characteristics of the genetic operators remain implementation dependent, such as whether both of the new structures obtained from crossover are retained, whether the parents themselves survive, and which other knowledge structures are replaced if the population size is to remain constant. In addition, issues such as the size of the population, crossover rate, mutation rate, generation gap, and selection strategy have been shown to affect the efficiency with which a genetic algorithm operates [46]

The inherent power of a genetic algorithm lies in its ability to exploit, in a highly efficient manner, information about a large number of individuals. By allocating more reproductive occurrences to above average individuals, the overall net affect is an upward shift in the population's average fitness. Since the overall average moves upward over time, the genetic algorithm is a "global force" which shifts attention to productive regions (groups of highly fit individuals) in the search space. However, since the population is distributed throughout the search space, genetic algorithms effectively minimize the problem of converging to local maxima.

To date, genetic algorithms have been applied to a wide diversity of problems. They have been used in combinatorial optimization [55], gas pipeline operations [42, 45] and machine learning[53]. With regards to computer vision applications, Mandava et. al [64] have used genetic algorithms for image registration, Gillies [41], and Roth and Levine [91] for feature extraction, and Ravichandran [85] for object recognition.

2.3 Genetic Learning for Adaptive Image Segmentation

Genetic algorithms can be used in three different ways to provide an adaptive behavior within a computer vision system. The simplest approach is to allow the genetic system to modify a set of control parameters that affect the output of an existing computer vision program. By monitoring the quality of the resulting program output, the genetic system can dynamically change the parameters to achieve the best performance. A second approach allows the genetic component to modify the complex data structures within an algorithm or production rule system for a computer vision application. By modifying the control mechanism or agenda in an algorithm or the organization of data frames in a rule-based system, the genetic algorithm can bring about changes in the system's behavior. Finally, the most complex implementation of an adaptive computer vision system allows the genetic algorithm to actually make changes in the executable code of a program. In most of these cases, the adaptation involves changing the condition/action statements of the rules in a production system. Since almost every image segmentation algorithm contains parameters that are used to control the segmentation results, we have adopted the first strategy listed above.

The block diagram of our approach to adaptive image segmentation is shown in Fig. 2.2. After acquiring an input image, the system analyzes the image characteristics and passes this information, in conjunction with the observed external variables, to the genetic learning component. Using this data, the genetic learning system selects an appropriate parameter combination, which is passed to the image segmentation process. After the image has been segmented, the results are evaluated. If the quality of segmentation ("fitness") is acceptable, and update to long-term population is made. If the quality is unacceptable, the process of new parameter selection, segmentation and evaluation continues until a segmentation result of acceptable quality is produced, or the terminate criteria are satisfied.

2.3.1 Image Characteristics

A set of characteristics of the image is obtained by computing specific properties of the digital image itself as well as by observing the environmental conditions in which the image was acquired. Each type of information encapsulates knowledge that can be used to determine a set of appropriate starting points for the parameter adaptation process. For the experiments described here, we compute twelve first order properties for each color component (red, green, and blue) of the image. These features include mean, variance, skewness, kurtosis, energy, entropy, x intensity centroid, y intensity centroid, maximum peak height, maximum peak location, interval set score, and interval set size [60, 101]. The last two features measure histogram properties used directly by the *PHOENIX* segmenta-

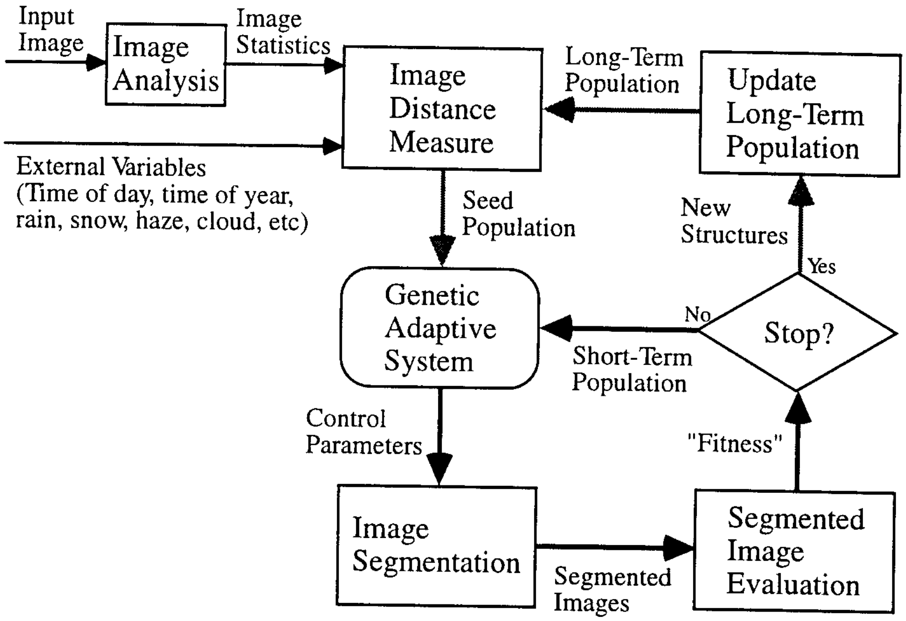


Figure 2.2: Adaptive image segmentation system.

tion algorithm and provide useful image similarity information. Since we use a gray scale image to compute edge information and object contrast during the evaluation process, we also compute the twelve features for the Y (luminance component) image as well. Combining the image characteristic data from these four components yields a list of 48 elements. In addition, we utilize two external variables, time of day and weather conditions, in the outdoor experiments to characterize each image. The external variables are represented symbolically in the list structure (e.g., time = 9am, 10am, etc. and weather conditions = sunny, cloudy, hazy, etc). The distances between these values are computed symbolically when measuring image similarity. The two external variables are added to the list to create an image characteristic list of 50 elements for the outdoor experiments. The representation of an individual knowledge structure of the genetic population is shown in Fig. 2.3.

2.3.2 Genetic Learning System

Once the image statistics and external variables have been obtained, the genetic learning component uses this information to select an initial set of segmentation algorithm param-

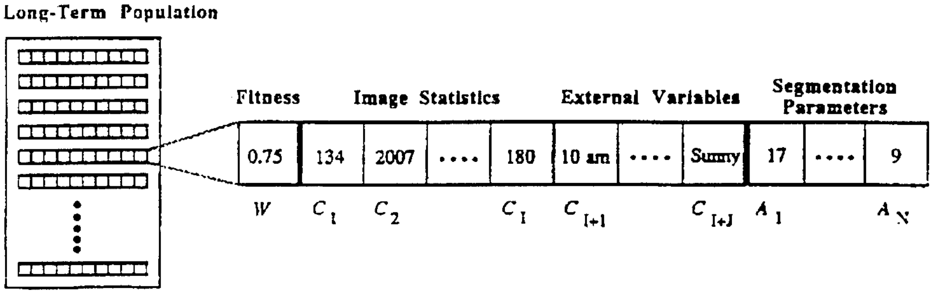


Figure 2.3: Representation of a knowledge structure used by the genetic learning system. The image characteristics (image statistics and external variables), segmentation parameters, and the image quality or fitness of the parameter set are stored in each structure.

eters. A knowledge-based system is used to represent the image characteristics and the associated segmentation parameters. The image statistics and external variables shown in Fig. 2.3 form the condition portion of the knowledge structure, C_1 through C_{I+J}, while the segmentation parameters indicate the actions, A_1 through A_N, of the knowledge structure. The fitness, W, which ranges in value from 0.0 to 1.0, measures the quality of the segmentation parameter set. Note that only the fitness value and the action portion of the knowledge structure are subject to genetic adaptation; the conditions remain fixed for the life of the knowledge structure.

When a new image is provided to the genetic learning system, the process begins by comparing the image characteristics of the new image (Fig. 2.2) with the knowledge structures in the long-term population (also called global population, Fig. 2.3). The long-term population represents the accumulated knowledge of the adaptive system obtained through previous segmentation experience. The algorithm computes a ranked list of individuals in the population that have characteristics similar to the new image. Ranking is based on the normalized Euclidean distance between the image characteristic values as well as the fitness of the knowledge structure. The normalized distance between images A and B is computed using

$$dist_{AB} = \sum_{i=1}^{I+J} W_i \left| \frac{C_{iA} - C_{iMIN}}{C_{iMAX} - C_{iMIN}} - \frac{C_{iB} - C_{iMIN}}{C_{iMAX} - C_{iMIN}} \right| ,$$

where C_{iMIN} is the minimum value of the ith numeric or symbolic feature in the global population, C_{iMAX} is the maximum value of the ith feature in the global population, and W_i is the weight attached to the ith feature. For the results presented in this paper, the ranges are normalized and the W_i values have been set to 1 so that each feature contributes equally to the distance calculation.

When the distance between an image and several members of the global population are the same (e.g., if a previous image contributed multiple individuals to the global population), fitness values are used to select the best individuals from the population. Temporary copies of the highest ranked individuals are used to create the initial or seed population for the new image.

Once the initial or seed population is available, the genetic adaptation cycle begins. (The seed population is the same as the initial population, when the genetic algorithm begins its search operation.) The segmentation parameter set in each member of the seed population is used to process the image. The quality of the segmented results for each parameter set is then evaluated. If the maximum segmentation quality for the current population is above a predefined threshold of acceptance or other stopping criteria are satisfied, the cycle terminates and the high quality members of the current image population are used to update the global population. Less fit members of the global population are discarded in favor of higher strength individuals obtained from processing the current image. In this manner, the system is able to extend the knowledge of the adaptive segmentation system by incorporating new experience into the knowledge database.

Alternatively, if after segmenting and evaluating the performance of the current or local (also called short-term) population, the system has not achieved acceptable segmentation quality and any other termination criteria are not satisfied, the genetic recombination operators are applied to the members of the current population. The crossover and mutation operators are applied to the high strength individuals in the population, creating a new set of offspring which will theoretically yield better performance [52]. The new population is supplied back to the image segmentation process, where the cycle begins again. Each pass through the loop (segmentation-evaluation-recombination) is known as a generation. The cycle shown continues until the maximum fitness achieved at the end of a generation exceeds some threshold or other termination criteria are satisfied, as described earlier. The global population is updated and the system is then ready to process a new image.

Fig. 2.4 provides a simple example of the adaptive segmentation system. The image characteristics extracted from the image are used in this example as the new image data. A subset of the complete image characteristics is used here for the sake of simplicity. The new image characteristics are compared with the individuals in the global population to obtain the seed population. The normalized Euclidean distance is computed from the new image to every member of the global population and this distance is used in conjunction with the fitness of each individual in the population. In this example, we have limited the seed population to 3 individuals. In the experiments described in Section 4, the seed population for each image consists of 10 knowledge structures while the global population holds 100 knowledge structures in order to maintain a diverse collection of segmentation experience. The matching process identifies the three members of the global population with similar image characteristics and a high degree of fitness. A copy of these individuals is then extracted to create the seed population shown in Fig. fig:gacycle.

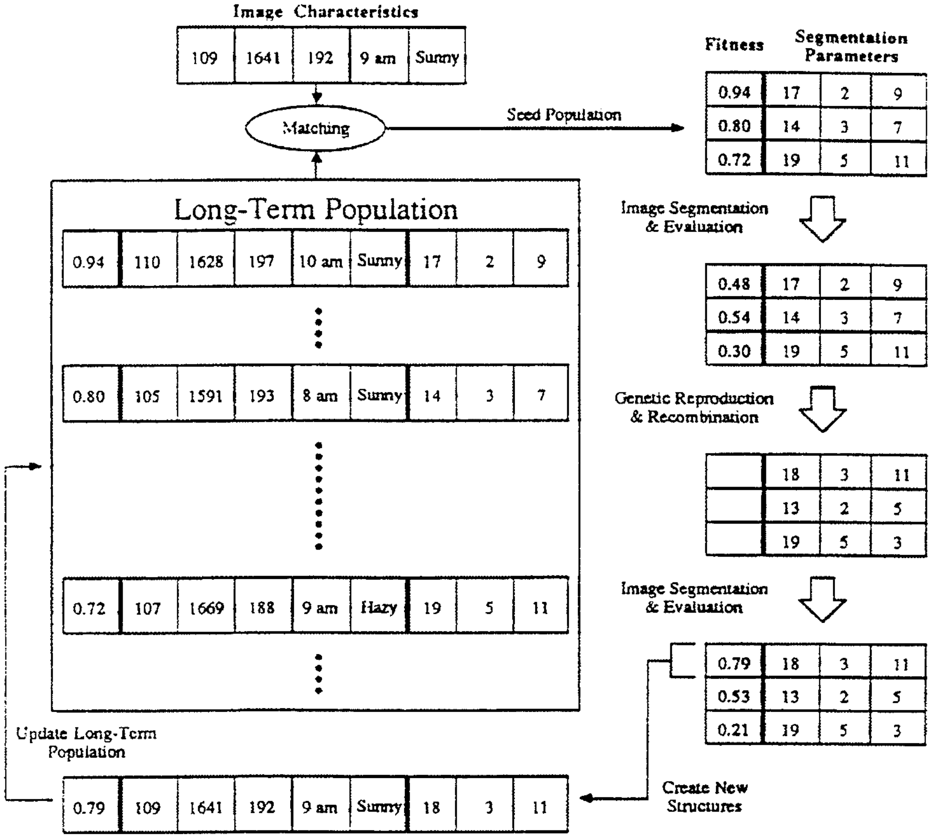


Image Characteristics
109 | 1641 | 192 | 9 am | Sunny
Matching
Seed Population
Fitness
Segmentation Parameters
0.94 | 17 | 2 | 9
0.80 | 14 | 3 | 7
0.72 | 19 | 5 | 11
Long-Term Population
0.94 | 110 | 1628 | 197 | 10 am | Sunny | 17 | 2 | 9
0.80 | 105 | 1591 | 193 | 8 am | Sunny | 14 | 3 | 7
0.72 | 107 | 1669 | 188 | 9 am | Hazy | 19 | 5 | 11
Image Segmentation & Evaluation
0.48 | 17 | 2 | 9
0.54 | 14 | 3 | 7
0.30 | 19 | 5 | 11
Genetic Reproduction & Recombination
18 | 3 | 11
13 | 2 | 5
19 | 5 | 3
Image Segmentation & Evaluation
0.79 | 18 | 3 | 11
0.53 | 13 | 2 | 5
0.21 | 19 | 5 | 3
Create New Structures
Update Long-Term Population
0.79 | 109 | 1641 | 192 | 9 am | Sunny | 18 | 3 | 11

Figure 2.4: Example of one complete cycle through the adaptive image segmentation system.

After the seed population is obtained, we must establish the fitness of each individual by processing the new image with the associated segmentation parameters. Thus, we perform the image segmentation and evaluation steps to derive the new fitness values shown in the second step of Fig. 2.4. Assuming a threshold fitness value of 0.70, none of the fitness values obtained from the initial population (0.48, 0.54, and 0.30) are acceptable. The current population is now passed through the genetic recombination step to generate a set of new individuals. In this example, the first two members of the population are combined using the crossover operator while the third member is modified using the mutation operator.

The newly created population must be passed through the segmentation and evaluation stages once more to determine the fitness of each individual. The fitness values of the new population are shown in Fig. 2.4. Since the fitness of the first individual (0.79) exceeds the threshold value (0.70), the adaptive cycle for this image is terminated. A new knowledge structure is created for the current image by inserting the appropriate image characteristics and storing the new parameter settings and their associated fitness. This knowledge structure is then inserted into the global population, replacing the least fit member. Had any other members of the new image's population been greater than the threshold, they too would have been placed into the global population.

2.3.3 Segmentation Algorithm

Since we are working with color imagery in our experiments, we have selected the *PHOENIX* segmentation algorithm developed at Carnegie-Mellon University and SRI International [60, 73, 101]. The *PHOENIX* algorithm is a recursive region splitting technique. An input image typically has red, green, and blue image planes, although monochrome images, texture planes, and other pixel-oriented data may also be used. Each of the data planes is called a feature or feature plane. The algorithm recursively splits nonuniform regions in the image into smaller subregions on the basis of a peak/valley analysis of the histograms of the red, green, and blue image components simultaneously. Segmentation begins with the entire image, considered to be a single region, based on histogram and spatial analyses. If the initial segmentation fails, the program terminates; otherwise, the program fetches each of the new regions in turn and attempts to segment them. This process terminates when the recursive segmentation reaches a predefined depth, or when all the regions have been segmented as finely as various user-specified parameters permit.

PHOENIX contains seventeen different control parameters [60] fourteen of which are used to control the thresholds and termination conditions of the algorithm. There are about 10^{40} conceivable parameter combinations using these fourteen values. For the outdoor image sequence that we have used, these parameters can be divided into three groups according to their effect on segmentation results.

Group I: Essential *PHOENIX* Parameters.

Parameter (default)	*Description*	*Range*
Hsmooth (9)	The width of the averaging window used to smooth each feature histogram.	1 – 100
Maxmin (160)	The minimum acceptable ratio of apex height to higher shoulder.	100 – 10^4

Group II: Important *PHOENIX* Parameters.

Parameter (default)	*Description*	*Range*
Absscore (70)	The lowest interval set score that will be passed to the threshold phase.	0 – 1000
Splitmin (4)	Direct manipulation of the segmentation queue, for which fetched regions are to be segmented further	1 – 200
Noise (10)	The size of the largest area that is to be considered noise	0 – 10^4
Height (20)	The minimum acceptable apex height as a percentage of the second highest apex	0 – 100

Group III: Less important *PHOENIX* parameters

The rest of the parameters have relatively much less influence on the segmentation result.

To minimize the problem complexity, four parameters have been selected for GA to search for the combination that gives best segmentation result using *PHOENIX*. Thirty two values are sampled for each of these four parameters. This results in a search space whose size is about one million. The parameters are shown in Table 1, together with the formula by which they are sampled, and the associated test range for each. In Section 4, we will present results using the first two parameters (*hsmooth* and *maxmin*). In Section 5, we show scaling results when we adapt all the four parameters.

2.3.4 Segmentation Evaluation

Table 2.1: *PHOENIX* parameters used for adaptive image segmentation.

Parameter	*Sampling Formula*	*Test Range*
Hsmooth: $hsindex \in [0:31]$	$hsmooth = 1 + 2 \cdot hsindex$	1 - 63
Maxmin: $mmindex \in [0:1]$	$ep = \log(100) + 0.05 \cdot mmindex$ $maxmin = \exp(ep) + 0.5$	100 - 471
Splitmin: $smindex \in [0:1]$	$splitmin = 9 + 2 \cdot smindex$	9 - 71
Height: $htindex \in [0:1]$	$height = 4 + 2 \cdot htindex$	4 - 66

After the image segmentation process has been completed by the *PHOENIX* algorithm, we must measure the overall quality of the segmented image. There are a large number of segmentation quality measures that have been suggested in the literature [11], although none has achieved widespread acceptance as a universal measure of segmentation quality. In order to overcome the drawbacks of using only a single quality measure, we have incorporated an evaluation technique that uses five different quality measures to determine the overall fitness for a particular parameter set. In the following, boundary pixels refer to the pixels along the borders of the segmented regions, while the edges obtained after applying an edge operator are called edge pixels. The five segmentation quality measures that we have selected are,

1. *Edge-Border Coincidence*: Measures the overlap of the region borders in the image acquired from the segmentation algorithm relative to the edges found using an edge operator. In this quality measure, we use the Sobel operator to compute the necessary edge information. The original, unthinned Sobel edge image is used to maximize overlap between the segmented image and the edge image. Edge-border coincidence is defined as follows (refer to Fig. 2.5(a)).

   Let E be the set of pixels extracted by the edge operator after thresholding and S be the set of pixels found on the region boundaries obtained from the segmentation algorithm:

$$E = \{p_1, p_2, \cdots, p_E\} = \{(x_{p1}, y_{p1}), (x_{p2}, y_{p2}), \cdots, (x_{pE}, y_{pE})\} \quad and$$

$$S = \{q_1, q_2, \cdots, q_S\} = \{(x_{q1}, y_{q1}), (x_{q2}, y_{q2}), \cdots, (x_{qS}, y_{qS}\}, \quad then$$

$$\text{Edge-border Coincidence} = \frac{n(E \cap S)}{n(E)}$$

$$E \cap S = \{(x_k, y_k), k = 1, \cdots, m, where(x_y, y_k) \in E \text{ and } S\}, \quad and$$

$$n(A) = \text{the number of elements in set A.}$$

2. *Boundary Consistency*: Similar to edge-border coincidence, except that region borders which do not exactly overlap edges can be matched with each other. In addition, region borders which do not match with any edges are used to penalize the segmentation quality. The Roberts edge operator is used to obtain the required edge information. As with the edge-border coincidence measure, the Roberts edge image is not thinned to maximize the overlap between images. Boundary consistency is computed in the following manner (see Fig. 2.5(b)).

   The first step is to find neighboring pixel pairs in the region boundary and edge results. For each pixel in the segmented image region boundary results, S, a neighboring pixel in the edge image, E, that is within a distance of d_{max} is sought. A reward for locating a neighbor of the ith boundary pixel is computed using

$$R_i = \frac{d_{max} - d_i}{d_{max}},$$

   where $d_{max} = 10$, and $d_i =$ the distance to the nearest edge pixel.

   Thus, if the pixels had overlapped, $R_i = (10 - 0)/10 = 1$. Pixels that do not directly overlap contribute a reward value that is inversely related to their distance from each other. As matching pairs of pixels are identified, they are removed from the region boundary and edge images (S and E). The total reward for all matching pixel pairs is obtained using

$$R_{TOTAL} = \sum_i R_i$$

   Once all neighboring pixel pairs have been removed from E and S, the remaining (i.e., non-overlapping and non-neighboring) pixels correspond to the difference between the two images. The average number of these pixels is used to compute a penalty

$$P = \frac{n(\text{all remaining pixels in } E \text{ and } S)}{2}.$$

   Finally, since the value of boundary discrepancy must be positive, we define an intermediate value, M, as $M = (R_{TOTAL} - P)/n(E)$, then

$$\text{Boundary Consistency} = M, \quad \text{if } M \geq 0, \quad \text{and zero otherwise.}$$

3. *Pixel Classification*: This measure is based on the number of object pixels classified as background pixels and the number of background pixels classified as object pixels. Let G be the set of object pixels in the groundtruth image and R be the set of object pixels in the segmented image (see Fig. 2.5(c)). Formally, we have

$$G = \{p_1, p_2, \cdots, p_A\} = \{(x_{p1}, y_{p1}), (x_{p2}, y_{p2}), \cdots, (x_{pA}, y_{pA})\} \quad \textit{and}$$
$$R = \{q_1, q_2, \cdots, q_B\} = \{(x_{q1}, y_{q1}), (x_{q2}, y_{q2}), \cdots, (x_{qB}, y_{qB}\}.$$

Since pixel classification must be positive, we define the intermediate value N as follows

$$N = 1 - \left[\frac{(n(G) - n(G \cap R)) + (n(R) - n(G \cap R))}{n(G)}\right]$$
$$\text{where } \; G \cap R = \{(x_k, y_k), k = 1, \cdots, m, \quad \text{where } \; (x_k, y_k) \in G \text{ and } R\}$$

Using the value of N, pixel classification can then be computed as

$$\text{Pixel Classification} = N, \quad \text{if } N \geq 0, \quad \text{and zero otherwise.}$$

4. *Object Overlap*: Measures the area of intersection between the object region in the groundtruth image and the segmented image, divided by the object region. As defined in the pixel classification quality measure, let G be the set of object pixels in the groundtruth image and R be the set of object pixels in the segmented image (Fig. 2.5(d)). Object overlap can be computed as

$$\text{Object Overlap} = \frac{n(G \cap R)}{n(G)}$$
$$\text{where } \; G \cap R = \{(x_k, y_k), k = 1, \cdots, m, \quad \text{where } \; (x_k, y_k) \in G \text{ and } R\}$$

5. *Object Contrast*: Measures the contrast between the object and the background in the segmented image, relative to the object contrast in the ground-truth image. Let G be the set of object pixels in the groundtruth image and R be the set of object pixels in the segmented image, as shown in Fig. 2.5(a). In addition, we define a bounding box (X and Y) for each object region in these images. These boxes are obtained by enlarging the size of the minimum bounding rectangle for each object (G and R) by 5 pixels on each side. The pixels in regions X and Y include all pixels inside these enlarged boxes with the exception of the pixels inside the G and R object regions. We compute the average intensity for each of the four regions (G, R, X, and Y) using the equation $I_L = \sum_{j=1}^{L_{max}} I(j)/L_{max}$, where I(j) is the intensity of the jth pixel in some region L and L_{max} is the total number of pixels in region L. The contrast of the

object in the groundtruth image, C_{GT}, and the contrast of the object in the segmented image, C_{SI}, can be computed using

$$C_{GT} = \left|\frac{I_G - I_X}{I_G}\right|, \quad C_{SI} = \left|\frac{I_R - I_Y}{I_R}\right|.$$

The object contrast quality measure is then computed as

$$\begin{aligned} \text{Object Contrast} &= \frac{C_{SI}}{C_{GT}}, \quad \text{if } C_{GT} \geq C_{SI} \\ &= \frac{C_{GT}}{C_{SI}}, \quad \text{if } C_{GT} < C_{SI}. \end{aligned}$$

The maximum and minimum values for each of the five segmentation quality measures are 1.0 and 0.0, respectively. The first two quality measures are *global* measures since they evaluate the segmentation quality of the whole image with respect to edge information. Conversely, the last three quality measures are *local* measures since they only evaluate the segmentation quality for the object regions of interest in the image. When an object is broken up into smaller parts during the segmentation process, only the largest region which overlaps the actual object in the image is used in computing the local quality measures. The three local measures require the availability of object groundtruth information in order to correctly evaluate segmentation quality. Since object groundtruth data may not always be available, we have designed the adaptive segmentation system to use three separate methods of evaluating segmentation quality. First, we can measure quality using global evaluation methods alone. Second, if groundtruth data is available and we are only interested in correctly segmenting the object regions in the image, we can use local evaluation methods alone. Finally, if we desire good object regions as well as high quality overall segmentation results, we can combine global and local quality measures to obtain a *combined* segmentation quality measure that maximizes overall performance of the system. In the experiments described in this chapter, we combine the five quality measures into a single, scalar measure of segmentation quality using a weighted sum approach. Each of the five measures is given equal weighting in the weighted sum. Elsewhere we have investigated a more complex vector evaluation approach that provides multidimensional feedback on segmentation quality [16, 17]

2.4 Segmentation Results

2.4.1 Segmentation Results Using Genetic Algorithm

The adaptive image segmentation consists of the following steps:

```
1.    Compute the image statistics.
2.    Generate an initial population.
3.    Segment the image using initial parameters.
4.    Compute the segmentation quality measures.
5.    WHILE not <stopping conditions> DO
5a.       select individuals using the reproduction operator
5b.       generate new population using the crossover
          and mutation operators
5c.       segment the image using new parameters
5d.       compute the segmentation quality measures
      END
6.    Update the knowledge base using the new knowledge structures.
```

We have tested the performance of the adaptive image segmentation system on a time sequence of outdoor images that contains variations in the position of the light source (sun) and the amount of light as well as changing environmental conditions. The outdoor image database consists of twenty frames captured using a JVC GXF700U color video camera. The images were collected approximately every 15 minutes over a 4 hour period. A representative subset of these images is shown in Fig. 2.6. The original images were digitized to be 480×480 pixels in size but were subsequently subsampled (average of 4×4 pixel neighborhood) to produce 120×120 pixel images for the segmentation experiments. Weather conditions in our image database varied from bright sun to overcast skies. Varying light level is the most prominent change throughout the image sequence. The changing environmental conditions caused by movement of the sun also created varying object highlights, moving shadows, and many subtle contrast changes between the objects in the image. Also, the colors of most objects in the image are subdued. The car in the image is the object of interest. The auto-iris mechanism in the camera was functioning, which causes a similar appearance in the background foliage throughout the image sequence. Even with the auto-iris capability built into the camera, there is still a wide variation in image characteristics across the image sequence. This variation requires the use of an adaptive segmentation approach to compensate for these changes.

To precisely evaluate the effectiveness of the adaptive image segmentation system, we exhaustively defined the segmentation quality surfaces for each frame in the database. The car in the image is the object of interest for the pixel classification, object overlap, and object contrast segmentation quality measures. The groundtruth image for the car was obtained by manual segmentation of Frame 1 only for the image sequence. The Sobel and Roberts edge operator results, which are used in the computation of the edge-border coincidence and boundary consistency measures respectively, are obtained from the gray scale image (Y component of the YIQ image set) for each frame [18]. For the determination of object contrast, we used 5 pixels beyond the Minimum Bounding Rectangle (MBR) for each object region. For the results presented here, the *maxmin* and *hsmooth* parameters of the *PHOENIX* algorithm were used to control the segmentation quality and the segmentation quality surfaces were defined for preselected ranges of these two parameters as shown in Table 1. All the parameters that were not optimized were set at the default *PHOENIX* parameter values. These parameters remain fixed throughout all the experiments. By selecting 32 discrete values (5 bits of resolution) for each of these parameter ranges, the search space contained 1024 different parameter combinations. Fig. 2.7 presents the five individual segmentation quality surfaces and the combined surface for Frame 1 of the database. Notice that the surfaces are complex and hence, would pose significant problems to traditional optimization techniques.

The genetic component used a local or seed population size of 10, a crossover rate of 0.8, and mutation rate of 0.01. A crossover rate of 0.8 indicates that, on average, 8 out of 10 members of the population will be selected for recombination during each generation. The mutation rate of 0.01 implies that on average, 1 out of 100 bits is mutated during the crossover operation to insure diversity in the local population. The stopping criteria for the genetic process contains three tests. First, since the global maximum for each segmentation quality surface was known *a priori* (recall that the entire surface was precomputed), the first stopping criteria is the location of a parameter combination that produces quality of 95% or higher. In experiments where the entire surface is not precomputed, this stopping criteria would be discarded. Second, the process terminates if three consecutive generations produce a decrease in the average population fitness for the local population. Third, if five consecutive generations fail to produce a new maximum value for the average population fitness, the genetic process terminates. If any one of these three conditions is met, the processing of the current image is stopped and the maximum segmentation quality currently in the local population is reported.

Numerous experiments [16] were performed for training and testing to measure the optimization capabilities of the genetic algorithm and to evaluate the reduction in effort achieved by utilizing previous segmentation experience. In the following we present some of these results.

2.4.2 Performance Comparison with Other Techniques

Since there are no other known adaptive segmentation techniques with a learning capability in both the computer vision and neural networks fields to compare our system with, we measured the performance of the adaptive image segmentation system relative to the set of default *PHOENIX* segmentation parameters [60, 101] and a traditional optimization approach. The default parameters have been suggested after extensive amounts of testing by researchers who developed the *PHOENIX* algorithm [60]. The parameters for the traditional approach are obtained by manually optimizing the segmentation algorithm on the first image in the database and then utilizing that parameter set for the remainder of the experiments. This approach to segmentation quality optimization is currently a standard practice in state-of-the-art computer vision systems. Fig. 2.8 illustrates the quality of the segmentation results for Frames 1 and 11 using the default parameters and the traditional approach and contrasts this performance with our adaptive segmentation technique. Each result corresponds to the average segmentation performance produced by each technique for the first frame in the outdoor image database. By comparing the extracted car region in each of these images, as well as the overall segmentation of the entire image, it is clear that the adaptive segmentation results are superior to the other methods. For Frame 1 using the traditional approach, the segmentation quality is initially 95%, which is close to the adaptive segmentation quality. This value indicates that our segmentation evaluation measures are providing information similar to human perceptual performance.

The average segmentation quality for the adaptive segmentation technique is 95.8%. In contrast, the performance of the default parameters is only 55.6% while the traditional approach has a 63.2% accuracy. The outdoor experiments described above were conducted in a parallel fashion, i.e., all training and all testing was performed without the aid of previous segmentation experience. Although the testing experiments used the knowledge acquired during training, the tests were still performed in parallel. None of the segmentation experience obtained during testing was applied to subsequent testing images. Using multiple day experiments, we show that experience can be used to improve the segmentation quality over time.

The size of the search space in these experiments is 1024, since each of the two *PHOENIX* parameters are represented using 5 bits. The price paid for achieving consistent higher quality of segmentation is the average number of times (2.5) one has to go through the genetic loop. Thus, only 2.4% of the search space is explored to achieve the global maximum. The superiority of the results is not because of the ground-truth information but because of the power of the adaptive image segmentation system. Many additional tests, including the comparison with random walk approach are performed, that demonstrate the effectiveness of the reproduction and crossover operators [16].

2.4.3 Demonstration of Learning Behavior

To measure the improvement in efficiency achieved by immediately reusing segmentation experience, we also conducted a set of experiments. These experiments were designed to investigate the reduction in computational effort obtained by processing the images in a sequential rather than parallel manner. All the parameters were set as mentioned above in this section. Three separate sequential experiments were performed. In each case, the order of the images presented to the adaptive image segmentation system was modified to determine the sensitivity of the sequential process to variations in the image sequences. The first test processed the outdoor images in their original order, i.e., Frames 1, 2, 3, ..., 20. The second test processed the odd numbered images first and then the even numbered images, i.e., Frames 1, 3, 5, ..., 19 followed by Frames 2, 4, ..., 20. This order was chosen so that we could compare the performance of the sequential processing with the parallel experiments performed earlier. Finally, the third test altered the sequence of images to simulate a multi-day scenario where the frequency of image collection decreases to approximately one hour. The order of the images in this test is 1, 5, 9, 12, 16, 20, 3, 7, 11, 14, 18, 2, 6, 10, 13, 17, 4, 8, 15, 19. Each group of images in the sequence of Frames (1, 5, 9, 12, 16, 20), (3, 7, 11, 14, 18), (2, 6, 10, 13, 17), or (4, 8, 15, 19) was designed to represent a collection of images acquired on a different day. Thus, using the sequence of images described above, we have simulated a four day long collection of images.

For each of the three tests, the genetic population of the first frame in the image sequence was randomly selected. Once the segmentation performance for that frame was optimized by the genetic algorithm, the final population from that image was used to create the initial global population. This global population was then used to select the seed population for subsequent frames in the image sequence. The global population size was set to 100 for these experiments to insure a diversity of segmentation experience in the population. While the size of the global population remained below 100 members (prior to processing 10 frames), the final populations for each image were merely added to the current global population. After the size of the global population reached 100 individuals, the final populations from each successive image had to compete with the current members of the global population. This competition was based on the fitness of the individuals; highly fit members of a new local population replaced less fit members of the global population, thus keeping the size of the global population constant. Fig. 2.9 presents the performance results achieved by the adaptive image segmentation system during each of the three sequential tests.

Single Day Sequential Test Fig. 2.9(a) illustrates the performance of the system for the single day sequence (first test). The number of generations for the first frame is quite large since we started from a random collection of search points. The experience gained in processing the first frame is immediately utilized during the second frame. The number of generations has been reduced from 12 to 3. Similarly, for Frames 3 and 4, the number of generations decreases each time. Although the number of generations does increase at several points beyond the fourth frame, the overall trend of this plot does indicate a reduction in computational effort. This claim is evident by noting that for the 20 frames of outdoor imagery in this sequence, the adaptive image segmentation system optimizes the segmentation quality of 50% (10 out of 20) of these images using the information present in the global population. No iterations of the genetic generations are necessary in these cases.

Odd-Even Image Sequence Test Fig. 2.9(b) provides similar evidence of learning and computational savings for the sequence of images used in the second test. Note that the initial slope of the graph in this figure is not as steep as in Fig. 2.9(a). This difference is due to the fact that the image intervals have increased in this experiment (e.g., we take every other image instead of every image). Thus, the knowledge previously acquired by the adaptive process is not as immediately relevant to subsequent images as it was during the first test. However, once we have processed all odd numbered images, the number of generations required during the even numbered images is substantially smaller. It is interesting to note that the even numbered images which require several generations (Frames 6, 14, and 18) in this test also required similar efforts in the first test (Fig. 2.9(a)). This correlation implies that the knowledge currently in the global population was not sufficient to optimize the segmentation quality of these images without some assistance from the genetic algorithm. Finally, note that as was the case in the first test, the adaptive image segmentation system optimizes the segmentation quality of half the image sequence (10 of 20 frames) without invoking the genetic process.

Multiple Day Sequential Test Fig. 2.9(c) presents the computational efforts required for the multi-day simulation in the third test. Once again, we can see the difference in the initial slope of the graph, which is due to the order in which the images are encountered. In this case, since there is an even wider separation between the images than in the two previous tests, the number of generations required for the first few frames is much higher. Additionally, with the exception of some local irregularities, the graph in Fig. 2.9(c) shows the cyclical nature of the multi-day process. The irregularities are attributed to the troublesome frames (6, 14, and 18) described earlier. The images in the first "day" (frames 1, 5, 9, 12, 16, 20) show a continually decreasing level of computational effort. When the second sequence (frames 3, 7, 11, 14, 18) is encountered, the effort increases temporarily as the adaptive process fills in the knowledge gaps present as a result of the differences between the images in each sequence. The image sequence for the third "day" (frames 2, 6, 10, 13, 17) was handled with almost no effort by the genetic learning. Finally, the fourth image sequence (frames 4, 8, 15, 19) requires no effort by the genetic learning at all; each image is optimized by the information stored in the global population. Note that the third test contains the largest number of frames processed with no help from the genetic algorithm. Twelve of the twenty frames in this test were optimized using the global population.

2.5 Scaling the Number of Parameters

For the results presented in Section 4, we selected only two (*hsmooth* and *maxmin*) parameters of the *PHOENIX* algorithm. In this section, we present experimental details when we select four parameters (*hsmooth*, *maxmin*, *splitmin* and *height*) for adaptive image segmentation. In this case the size of the search space is about 1 million. Table 1 shows the parameter values. As the number of segmentation parameters for adaptation increases, the number of points to be visited on the surface will also increase. However, genetic algorithms offer a number of advantages over other search techniques. These include parallel search from a set of points with the *expectation* of achieving the global maximum. Unlike the Hough transform [9], which is essentially an exhaustive search technique commonly used in Computer Vision, it is expected that the genetic algorithm will visit only a small percentage of the search space to find an adequate solution, that is sufficiently close to the global maximum.

2.5.1 Search Space and GA Control Mechanism

Visualization of the Search Space Visualization of the search space allows one to understand its complexity—the number and distribution of local peaks and the location of global maximum. But this 5-dimensional space (four parameters plus the fitness or quality of image segmentation) is difficult to be visualized with traditional methods. So we project this 5-dimensional data into a 4-dimensional space by slicing it into 32 pieces along the *Height* axis.

Fig. 2.10 shows the 3-D volume representation of this 4-dimensional data using the brick and slice visualization technique, where the x, y, z axes are *maxmin*, *hsmooth*, and *splitmin* respectively (Fig. 2.11), and the color associated with each point represents the combined segmentation quality for a given parameter set. Blue color represents segmentation quality of zero, while the red color represents 100% quality. To create the data shown in this figure using *PHOENIX* took a couple of weeks on 10 SUN Sparc2 machines.

GA Control Mechanism As discussed earlier, GA require three operations: selection, crossover, and mutation. Here each chromosome consists of four parameters. The ordering of these parameters within the chromosome representation does not affect the search process due to our method of crossover point selection. Tests are carried out to select the best control parameters for GA. These include number of crossover points, crossover rate, mutation rate, method of selection, population size, and quality threshold. The results are given below.

Crossover Rate Table 2 shows the number of segmentations that are needed for frame 1 for different crossover rates. The threshold for minimum acceptable segmentation quality is 95%, population size varies from 50 to 200. We can see that a lower crossover rate leads to smaller number of total segmentations. These data are averaged over 1000 independent tests.

Table 2.2: Number of segmentations under varying population size and crossover rate. The threshold for minimum acceptable segmentation quality was set at 95%

Population	Crossover Rate	2-Point Crossover
50	80%	9439
	50%	6077
100	80%	5805
	50%	4675
200	80%	7548
	50%	5068

Table 2.3: Number of segmentations under varying population size and selection of crossover points. (Segmentation Quality Threshold = 95% , Crossover Rate = 80%).

Population	1-Point Crossover	2-Point Crossover	4-Point Crossover
10	7102	6553	5941
100	4960	5805	5528
200	4131	3939	3900
500	3575	3332	2878

Table 2.4: Number of segmentations under varying threshold (Population = 500, Crossover Rate = 80%).

Threshold	1-Point Crossover	2-Point Crossover	4-Point Crossover
95%	3575	3332	2878
90%	2943	2788	2325

Population Size and Number of Crossover Points Table 3 shows the number of segmentations required for different population sizes and crossover points. The threshold for acceptance of segmentation quality is 95% and the crossover rate is set at 80%. From the results we can see that using more crossover points and larger population size, the total number of required segmentations can be reduced. This experiment also showed that the total number of segmentations will not reduce further when population size is greater than 500. A complete scenario for crossover operation using four points is shown in Fig. 2.12.

Segmentation Quality Threshold Table 4 shows how different thresholds which correspond to minimum acceptable segmentation quality affect the total number of required segmentations. The difference is not significant between 90% and 95% because these segmentation qualities are quite close.

The results presented for Frame 1 in Tables 2-4 show that the number of points that are visited on the surface varies from 0.9% to 0.3% for 95% quality of segmentation. In the best case only 0.28% of the search space is visited to achieve 99.89% (Threshold is 95%) quality of segmentation.

2.5.2 GA Plus Hill Climbing for Adaptive Image Segmentation

Hybrid search techniques [1] have the potential for improved performance over single optimization techniques since these can exploit the strengths of the individual approaches in a cooperative manner. One such hybrid scheme which we describe in this section combines a global search technique (genetic algorithm) with a specialized local search technique (hill climbing). Hill climbing methods are not suitable for optimization of multimodal objective functions, such as the segmentation quality surfaces, since they only lead to local extrema and their applicability depends on the shape of the objective functions. The hybrid scheme provides performance improvements over the genetic algorithm alone by taking advantage of both the genetic algorithm's global search ability and the hill climbing's local convergence ability. In a sense, the genetic algorithm first finds the hills and the hill climber climbs them.

The search through a space of parameter values using hill climbing consists of the following steps: (1) Select a starting point; (2) Take a step in each of the fixed set of directions; (3) Move to the best alternative found; and (4) Repeat until a point is reached that is higher than all of its adjacent points. An algorithmic description of the hill climbing process is as follows:

```
1a.     Select a point $x_c$ at random.
1b.     Evaluate the criterion function, i.e., obtain $V(x_c)$
2a.     Identify points $x_1, \cdots, x_n$ adjacent to $x_c$
2b.     Evaluate the criterion function, i.e., obtain $V(x_1), \cdots, V(x_n)$.
3.      Let $V(x_m)$ be the maximum of $V(x_i)$ for $i = 1, \cdots, n$.
3a.     If $V(x_m) > V(x_c)$ then
            set $x_c = x_m, V(x_c) = V(x_m)$
            goto Step 2.
3b.     Otherwise, stop.
```

In the above, a set of points that are "adjacent" to a certain point can be defined in two ways. First, it can denote the set of points that are a Euclidean distance apart from the given point. Thus, the adjacent points are located in the neighborhood of the given point. Second, "adjacent" points can denote the set of points that are unit Hamming distance apart from the given point pair. Each point in this set differs by only one bit value from the given point in binary representation of points. It defines the set of points with varying step size from the given point. The set of Hamming adjacent points was used in this research. Hamming adjacent points have an advantage over Euclidean adjacent points in our implementation because all the segmentation parameter values are represented as binary strings when using the GA. The set of Hamming adjacent points also represents the set of points which can be generated by a genetic mutation operator from the given point.

A conventional hill climbing approach, as described above, finds the largest $V(x_m)$ from $V(x_i), i = 1, \cdots, n$, and the search moves to its corresponding point, x_m. For a space of n adjacent points, it requires n function evaluations to make each move. To reduce the cost of evaluating all the adjacent points before making each move, the hybrid approach is designed to try alternatives only until an uphill move is found. The first uphill move is undertaken without checking whether there are other (higher) possible moves. After the hill climbing process has examined all the adjacent points by flipping each bit in the binary representation of the current point, in turn, without finding an uphill move, the current point is taken as a local maximum. The algorithmic description of the hill climbing process used in the hybrid search scheme is as follows:

```
1.    Select a starting point $x_c$ with fitness value $V(x_c)$ from the
      genetic population.
2.    Set $i = 0$.
3.    Set $j = i$.
4a.   Generate an adjacent point $x_a$ by flipping the $i$th bit in $x_c$.
4b.   Obtain $V(x_a)$. Set $i = (i + 1) \bmod n$.
5.    If $V(x_a) > V(x_c)$ then
              set $x_c = x_a$
              goto Step 3.
      Else if $i < j$ then
              goto Step 4
      Otherwise, pass the control to the GA.
```

2.5.3 Experimental Results

There are several possibilities in which GA plus climbing can be used. In one case the control moves back and forth between GA and hill climbing [16, 17]. In the approach used here GA is used for obtaining starting points for hill climbing for the first frame only. Thereafter, only hill climbing is used.

1. *GA learning:* Perform GA learning for frame 1 using a population size of 10 (chosen from hardware consideration) and 4 point crossover operation with a crossover rate of 0.8 (same as in Section 4). Ten knowledge structures are selected as seeds for hill climbing. The goal here is to use small population size to achieve the desired segmentation quality with minimum number of segmentations.

2. *Hill climbing:* For frame 2 to frame 20 perform hill climbing with accumulated knowledge structures. The seeds generated from previous frames are used to hill climb. The best result obtained for the current frame is kept as a new knowledge structure and added to the seed pool for hill climbing for the next frame.

After we are done with frame 20, we will accumulate 29 knowledge structures, with 19 of them generated by hill climbing.

The experimental results of the hybrid search scheme (combining GA and hill climbing) for frame 1 are shown in Table 5. The results show that for 95% threshold for image segmentation quality, genetic algorithm plus hill climbing technique helps to reduce the

Table 2.5: Performance comparison between pure GA and GA with hill climbing (crossover points = 4, crossover rate = 80%, mutation rate ≈ 1%).

Population = 10	Genetic w/o hill climbing	Genetic with hill climbing
Threshold = 95%	5941	3340
Threshold = 90%	1720	1631

required number of segmentations by almost half. For low segmentation quality threshold (90%), this effect is not dramatic.

Fig. 2.13 summarizes the performance of GA plus hill climbing based techniques for frames 2 to 20, and compare it with default parameter set of the *PHOENIX* algorithm. The performance corresponds to the parameter set in the population that has the highest fitness. The average performance increase for the hybrid scheme over the default parameter set is about 50%, increase over the initial knowledge seed (GA learning for frame 1 only, no subsequent hill climbing) is also dramatic. This shows that GA learning from frame 1 does provide a good starting point for hill climbing. The average improvement shown in Fig. 2.13 is 107.8% over the default parameter set.

Figs. 2.14 and 2.15 compare the segmentation results obtained by using the default parameter set and the parameter set generated by GA and hill climbing. Using the default parameter set, it is seen that the car does not show up at all in the segmentation results for Frames 7 and 16, but the corresponding results using GA and hill climbing are quite good. The results show that by combining genetic search and hill climbing techniques the performance improvement is significant when the search space is large.

2.6 Conclusions

The goal of this research was to perform adaptive image segmentation and evaluate the convergence properties of the closed-loop system using outdoor data. The performance improvement provided by the adaptive system was consistently greater than 30% over the traditional approach or the default segmentation parameters [60, 101]. Further experimental details and several other techniques can be found in [16, 17, 18].

The adaptive image segmentation system can make use of any segmentation technique that can be controlled through parameter changes. No extensive knowledge pertaining to the selected algorithm is required. In addition, we can choose to adapt the entire parameter set or just a few of the critical parameters, depending on time constraints and the desired quality of the final segmentation results. The adaptive segmentation system is only as robust as the segmentation algorithm that is employed. It cannot cause an algorithm to modify the manner in which it performs the segmentation task. It can only optimize the manner in which the algorithm converges to its best solution for a particular image. However, it may be possible to keep multiple segmentation algorithms available and let the genetic process itself dynamically select the appropriate algorithm based on image characteristics. Further, it is possible to define various evaluation criteria which can be automatically selected and optimized in a complete vision system. Although we have only used color images in our current experiments, the adaptive technique itself is applicable to any type of imagery whose characteristics can properly be represented. This set includes infrared, laser radar, millimeter wave, sonar, and gray scale imagery. The adaptive image segmentation system may soon be able to benefit from advances in parallel computing and VLSI technology, which are now beginning to produce chips that can perform the image segmentation process in real time [13] These hardware improvements would make it possible to achieve high quality image segmentation results at near-realtime processing rates.

In a complete computer vision system, the segmentation evaluation component can be replaced by the object recognition component(for example, see [78]). In our adaptive image segmentation system, the focus is the image segmentation component. Therefore, we supplied the manually generated groundtruth image to the segmentation evaluation component and used local and global measures. Our approach attempts to fulfill the goal of automatic segmentation and groundtruth provides the reference against which the segmentation results can be evaluated. The groundtruth information is not contributing to the superiority of our approach since the same information is also being used by the traditional approach and the approach based on the default segmentation parameters that we have analyzed for performance comparison. Availability of such groundtruth information is guaranteed in such applications as photointerpretation and automatic target detection/recognition for the regions of interest containing the potential targets. The adaptive image segmentation system can utilize local, global, or combined segmentation quality measures to achieve the appropriate segmentation results. If nothing is known about an application, global evaluation measures can be used. For example, a complete target recognition system (in hardware) has been developed where edge/border coincidence [11] has been used for terminating image segmentation on real FLIR images. Elsewhere, we have optimized both global and local measures in a multi-objective optimization framework [17]. In the future we plan to use a data set with dramatic environmental variations and we will utilize several segmentation algorithms. Ultimately, we will incorporate the adaptive segmentation component into our complete vision system.

2.7 Future Work

- *Adaptive Multisensor Image Segmentation*

  Sensors are known to be sensitive to the scene content, such as the different channels of a multi-spectral scanner. Selection of sensors in a dynamic and adaptive fashion is therefore necessary to achieve the best possible segmentation results for a given scene. In this task, we will develop the mechanisms for selecting the most suitable sensor (i.e., one) from a set of sensors with identical imaging geometries (such as the channels of a multi-spectral scanner) as well as from sensors with different geometries and principles of operation (TV, FLIR, LADAR, etc.). The objective here is to integrate multiple sensors in a cooperative, complementary, and competitive fashion. The integrated results are evaluated and a GA-based approach is used to learn the appropriate strategy for combining the individual sensor data in order to obtain the best overall segmentation results. The combined data are fed back to the individual adaptive segmentation modules to enhance their performances further.

- *New evaluation criteria* – An important factor related to GA applications concerns the validity of the evaluation function used to evaluate population members [31]. If the evaluation function does not provide a uniform measure of individual fitness, the performance of the search process will suffer since the system will be placing unrealistic confidence in the strength of certain individuals based on the misinformation of the evaluation data. In order to prevent the quality of the evaluation function from adversely affecting the overall performance, Schaffer and Grefenstette [95] have developed a method of using vector performance evaluation incorporating multiple forms of evaluation information instead of a simple, scalar measure of performance. Alternatively, in our initial work we have combined five separate quality measures using a weighted sum approach to provide a more powerful and uniform indication of an individual's fitness. Additionally, the latter approach also maintains a vector of performance measures (global and local segmentation quality measures) for each genetic structure. Our proposed new evaluation criteria will include a combination of gray scale, texture, and color features. In the case of a complete object recognition system, the following quantitative measures can be used: 1) number of target pixels misclassified with respect to the true target, 2) correlation coefficient and mean squared-error between the true and extracted objects, 3) object-to-background contrast, intensity difference and Bhattacharyya distance [23] between the true object and clutter objects used with thresholding (threshold fixed *a priori* or determined in a global, local or object adaptive manner), and 5) shape number that estimates the shape difference between the true and extracted targets. In the navigation scenario, where knowledge-based landmark recognition is often employed, access to landmark models may provide geometric and semantic features that will guide the segmentation evaluation. Once the set of evaluation criteria is selected, the overall performance measure will be a *vector* of all the individual measures. The key research issues are (a) the selection of the relative weights for the different local and global measures, (b) learning of these weights using GA, (c) relationships of the selected measures to the information content of an image, which may be measured in terms of entropy, structural similarity of a pixel to its neighbors, or co-occurrence matrices, and (d) the confirmation of the local measures by human perceptual measures.

- *Recursive GAs - Optimization of genetics with genetics* – In our initial work, the use of GA has been solely to manipulate strings that represent parameter (of the segmentation algorithm) combinations. In DeJong's study [29] of GAs in function optimization, he suggests that good genetic algorithm performance requires the selection of a moderate population size, a high crossover probability, and a low mutation probability. For example, a small population size will cause the GA to converge too quickly without adequately exposing the system to learning experiences. On the other hand, a large population size results in a longer waiting period for significant improvements in the learning behavior. However, a setting of GA parameters, such as population size, crossover rate, mutation rate, generation gap, and selection strategy is in general implementation dependent [46]. Other characteristics of the genetic operators also remain implementation dependent, such as whether both of the new structures obtained from crossover are retained, whether the parents themselves survive, and which structures are removed if the population size is to remain constant. The scope of a GA-based adaptive image segmentation technique can be extended by incorporating *recursive* GAs (RGAs) in place of the single-level simple GA (SGA). The objective of recursive GAs will be to learn the above mentioned variables such as population size of the SGA. Besides, RGAs can also be used to learn the relative weights of the evaluation criteria.

- *Feedback from Higher-level Processing for Criteria Selection* – In most typical approaches to lower level computer vision tasks, including segmentation, little attention has been paid to providing feedback from higher level processes. On the other hand, a feedback from a higher level process, such as object recognition, would undoubtedly help in making such lower level decisions as to which segmentation evaluation criteria are proving to be the most effective in recognizing an object in a particular scene. Our proposed adaptive approach will include such feedback connections from the high- as well as the intermediate-level processes.

- *Use of Classifier Systems* – The most common genetic-based machine learning GBML architecture is known as the *classifier system.* A classifier system is a machine learning system that learns syntactically simple string rules or *classifiers* in an arbitrary environment. It consists of three main components [43]: *rule* and *message* system, apportionment of *credit* system, and *genetic algorithm.* In our preliminary work, we have focused on the GA-component of a classifier system. The rule and message system-component is a special kind of production system in that it restricts its rules to fixed-length representations. Unlike traditional expert systems, in which serial rule activation is the norm, classifier systems use parallel rule activation. Assigning rewards to the rules between successive evaluations via competition and rule discovery is the task of the apportionment of credit system. As a result of credit assignment, only good rules survive. Finally, the GA is responsible for creating new rules through reproduction, crossover and mutation. So far, our effort in adaptive image segmentation has been confined to the low-level domain. On the other hand, when more higher-level decisions are brought in to affect the low-level optimization of the segmentation results, rules will prove to be effective in representing human-like knowledge. Besides, there are existing segmentation algorithms that rely on rule databases [62]. We, therefore, plan to make use of complete classifier systems in adaptive image segmentation.

- *Comparison of Genetic Algorithms and Evolutionary Strategies* – Like the GAs, the Evolutionary Strategies (ESs) are a class of algorithms designed to synthesize natural evolution as a means of solving parameter optimization problems [8]. Functionally, both GAs and ESs are similar in the overall process of generating new individuals from the existing population. However, the architectures of these two approaches are vastly different. During the selection process for genetic recombination in ESs, either the offsprings in a generation are selected or both the parents and the offsprings are chosen. The recombination process in the ESs, which is analogous to the crossover operation in the GAs, can vary from no recombination, i.e., only one parent is subjected to mutation in a generation, to global recombination, in which both parents contribute equally to the creation of an offspring. But the key difference between the two approaches is in the implementation of the mutation operator that causes occasional changes in the genetic structure of an individual. While the mutation rate is almost always determined by an exogenous heuristic in GAs (and in earlier versions of ESs), it is treated as a part of the genetic structure in ESs thereby subjecting it to the same genetic processes as the parameters themselves. This constitutes a two-level learning process in which not only the population adapts to the response surface of the objective function, but also the rate of adaptation is adjusted according to the surface topology. An additional strategy is also incorporated to handle situations when the mutations of the individuals are correlated; the optimum rate of progress is achieved under correlated mutation condition. In our approach to image segmentation, we therefore plan to investigate the relative merits of GAs and ESs.

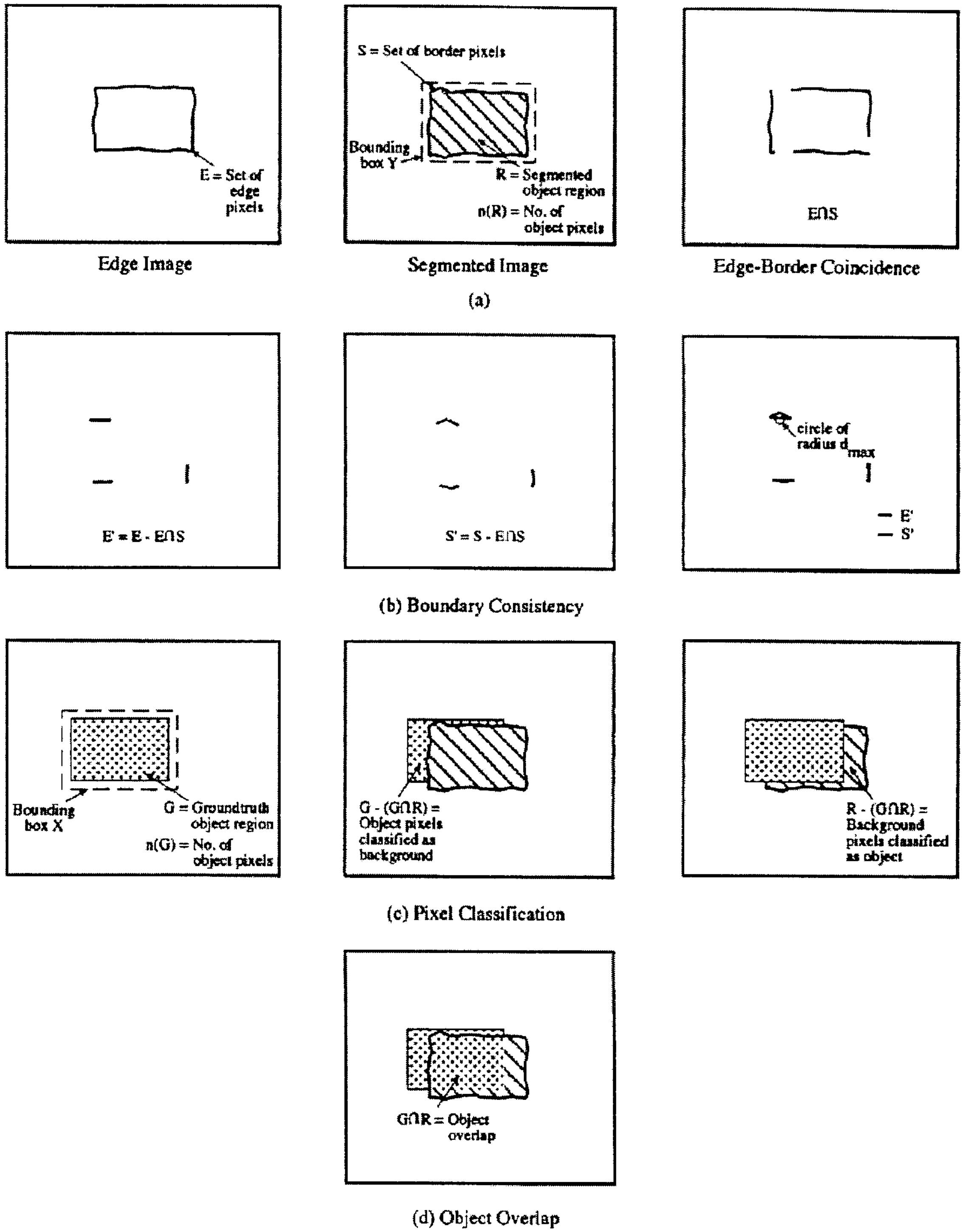
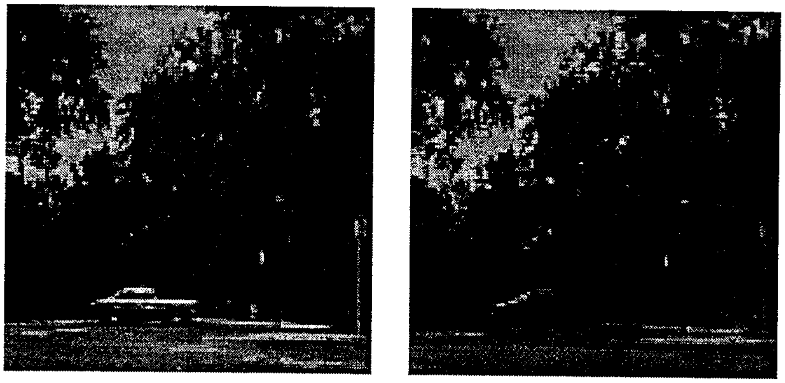


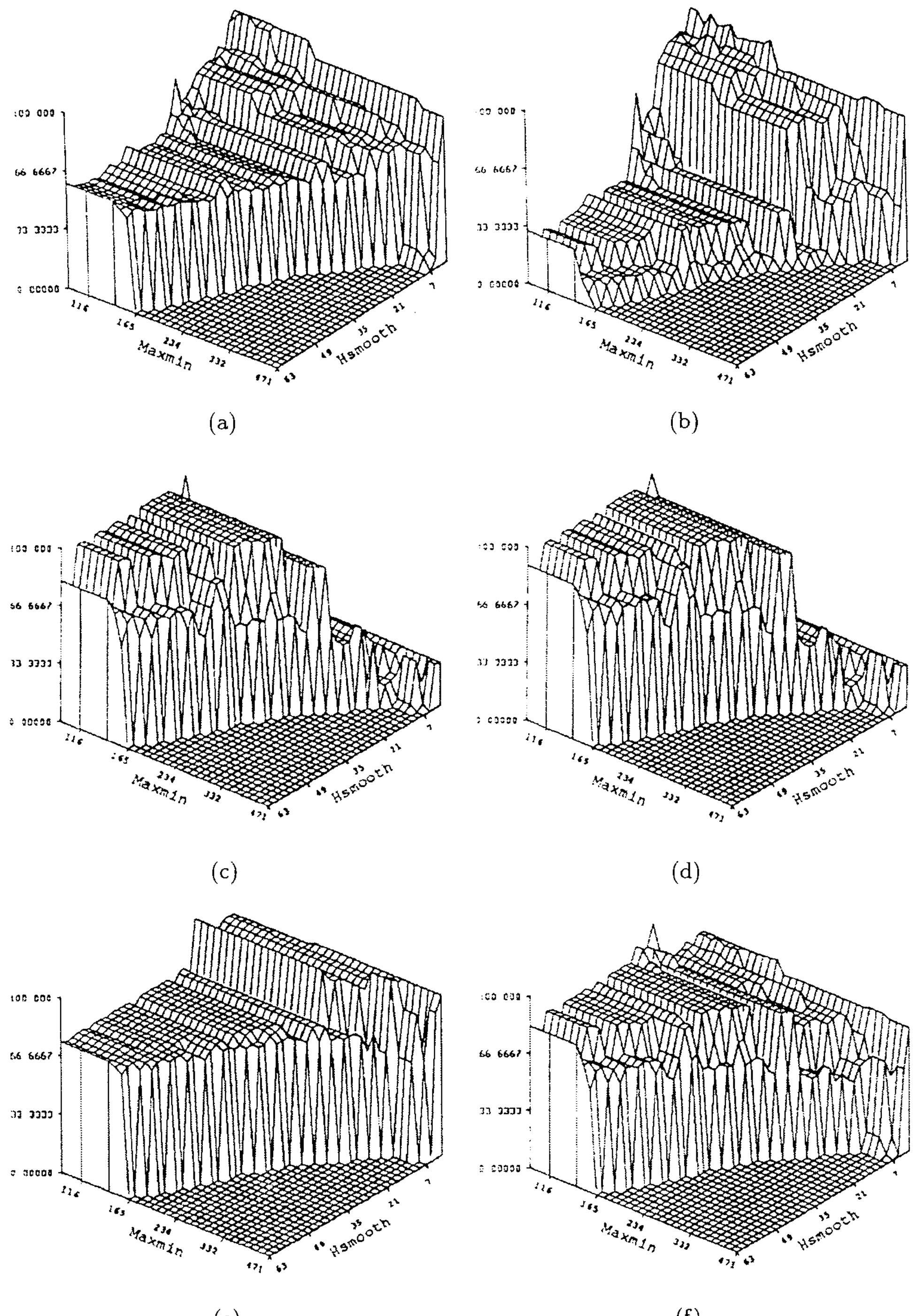
Figure 2.5: Illustration for the quality measures used in the adaptive image segmentation system. (a)Edge-border coincidence, (b)Boundary consistency, (c)Pixel classification, (d)Object overlap. Object contrast is defined by using the symbols shown in the center figure in (a) and the left most figure in (c).



(a) Frame 1

(b) Frame 11

Figure 2.6: Sample outdoor images used for adaptive segmentation experiments.



(a)

(b)

(c)

(d)

(e)

(f)

Figure 2.7: Segmentation quality surfaces for Frame 1. (a)Edge-border Coincidence, (b)Boundary Consistency, (c)Pixel Classification, (d)Object Overlap, (e)Object Contrast, (f)Combined Segmentation Quality.

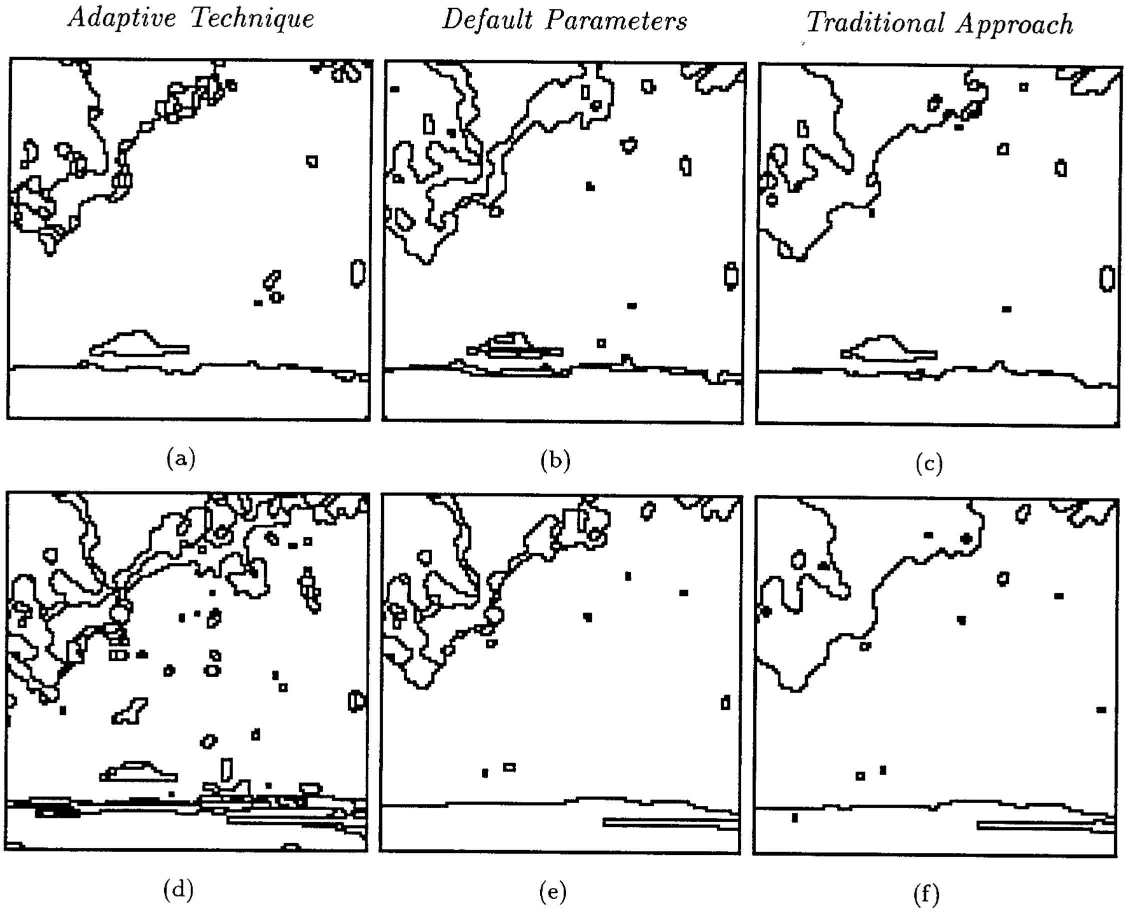


Figure 2.8: Segmentation of Frame 1 (a–c) and Frame 11 (d–f) for the adaptive technique, default parameters, and the traditional approach.

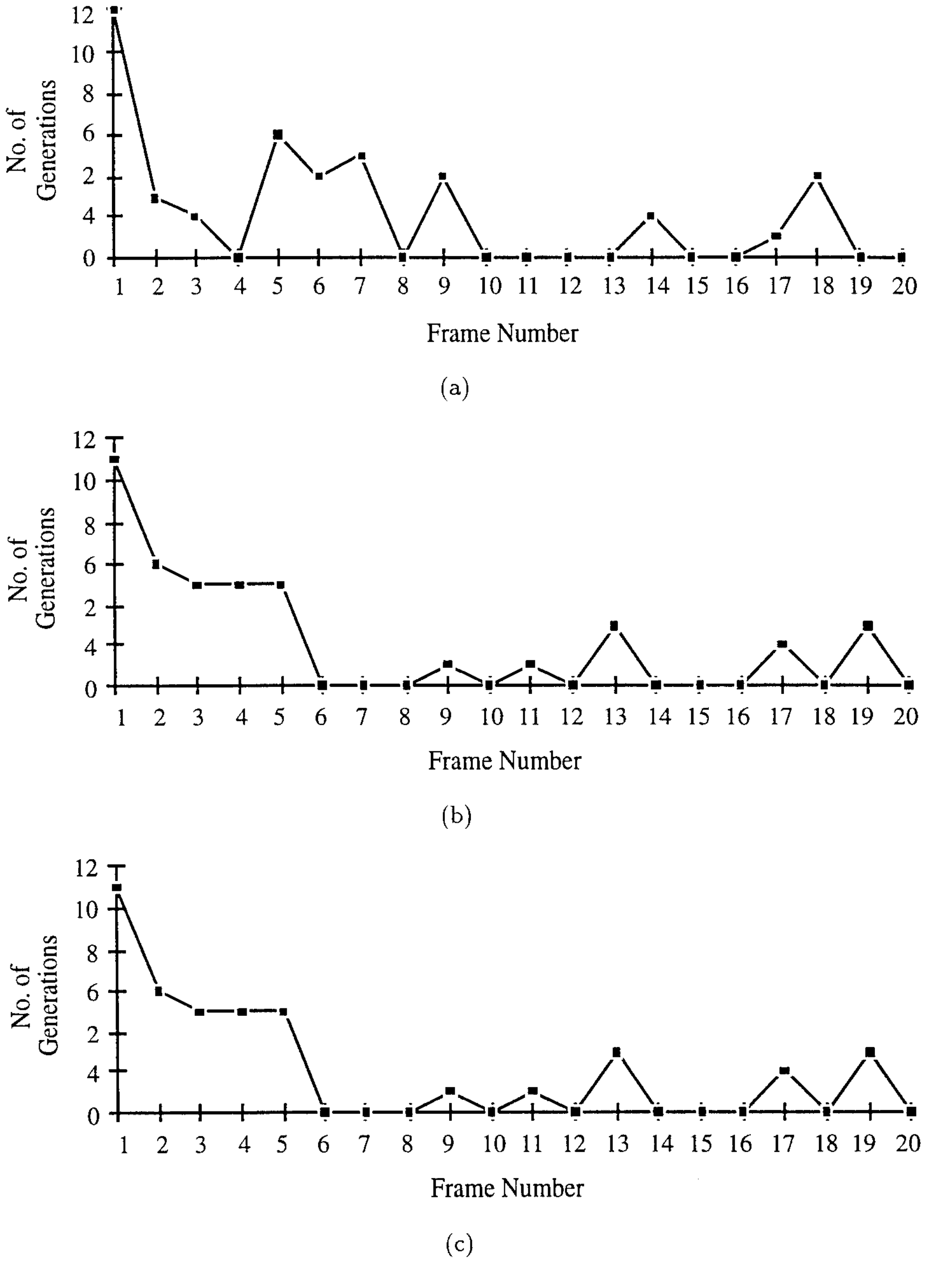


Figure 2.9: Performance of the adaptive image segmentation system for the sequential experiments. (a)Single day test results. (b)Double day test results. (c)Multiple day test results.

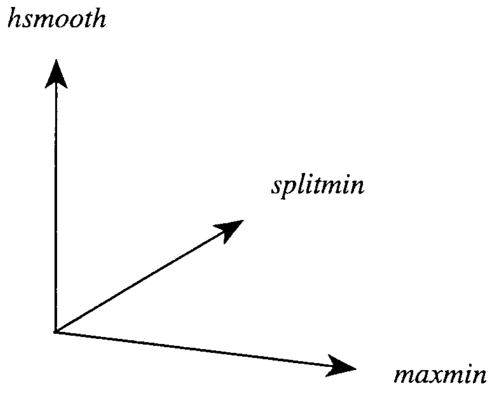
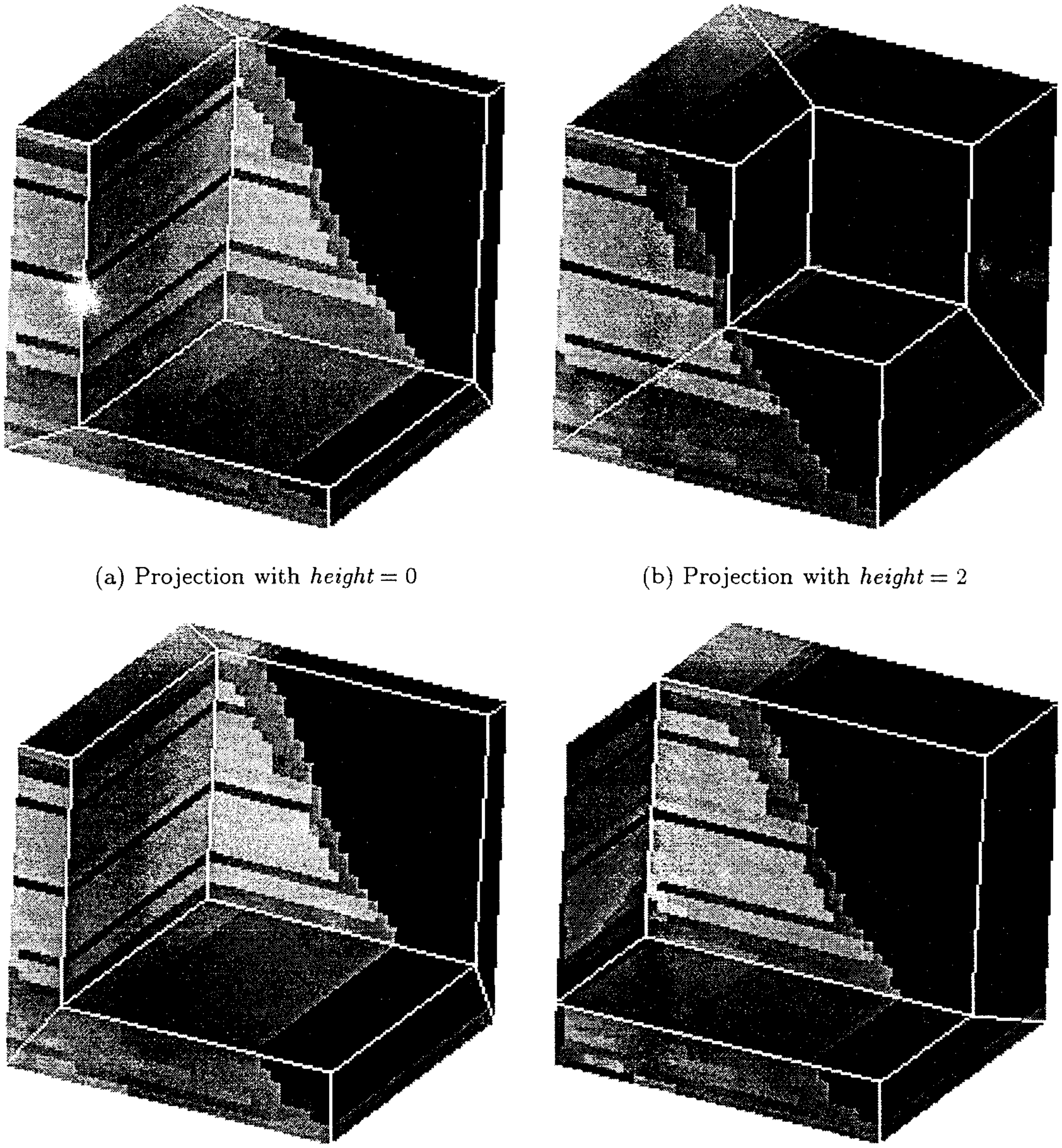


Figure 2.10: Coordinate axes for the volume representation in Fig. 10



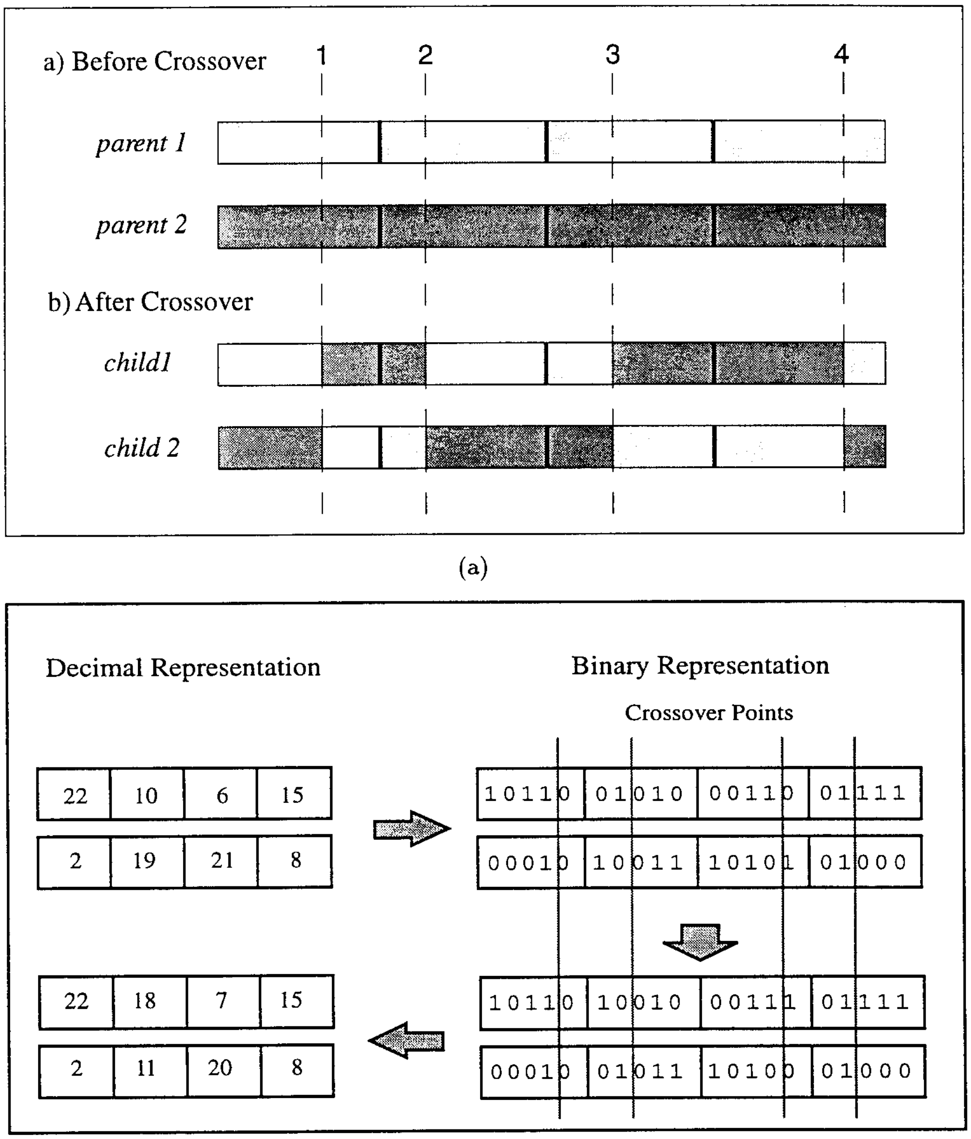
(a) Projection with *height* = 0

(b) Projection with *height* = 2

(c) Projection with *height* = 10

(d) Projection with *height* = 25

Figure 2.11: Volume representation (different views) of segmentation parameter search space. The original 5-dimensional data (*hsmooth*, *splitmin*, *maxmin*, *height*, segmentation quality) is projected along *height* axis, where the color represents the fitness or segmentation quality value corresponding to each 3-D coordinate.



(a)

(b)

Figure 2.12: Genetic algorithm crossover operation. (a) Scheme for doing 4-point crossover with each chromosome containing four parameters. (b) A complete scenario for one crossover operation.

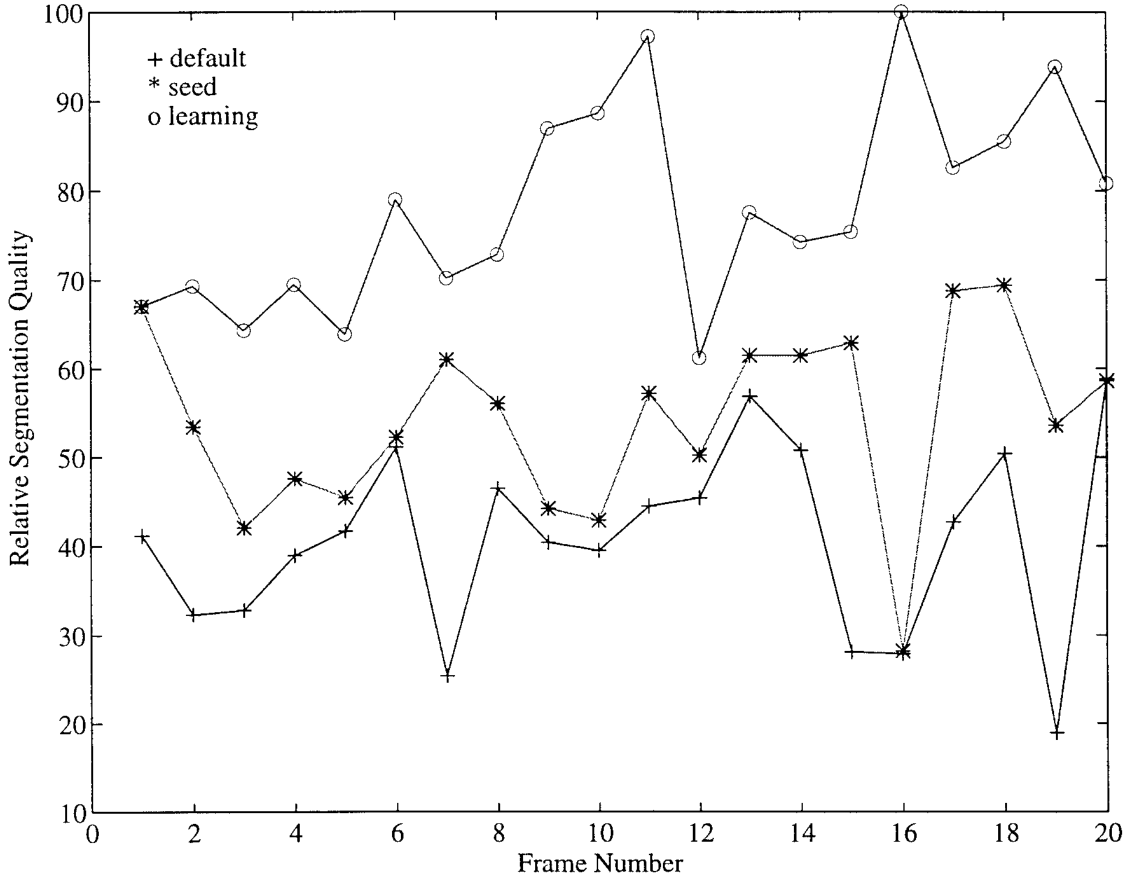
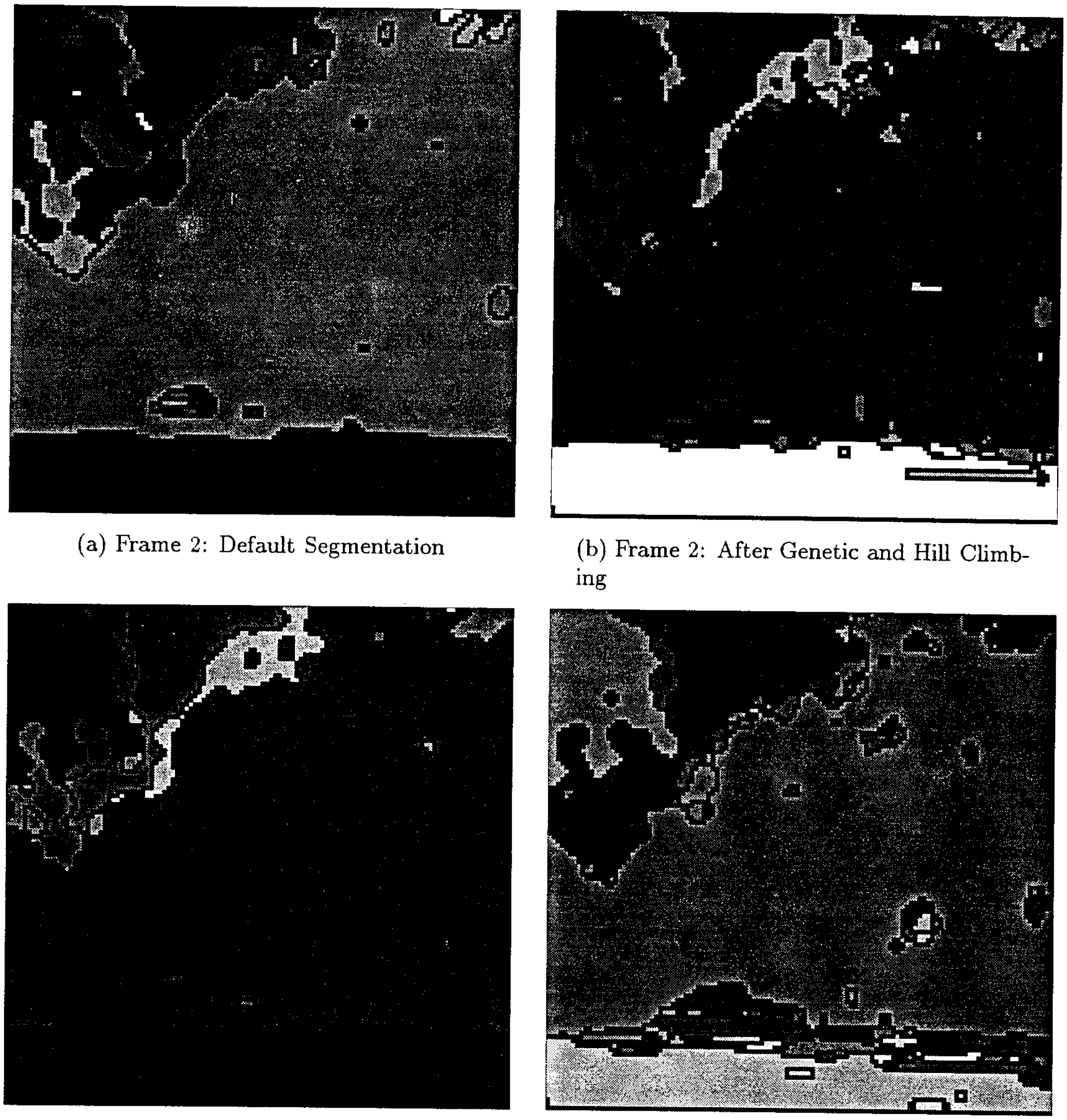


Figure 2.13: Performance comparison between default (+), initial seed (*), and final hill climbing (o) results for frame 1 to 20.



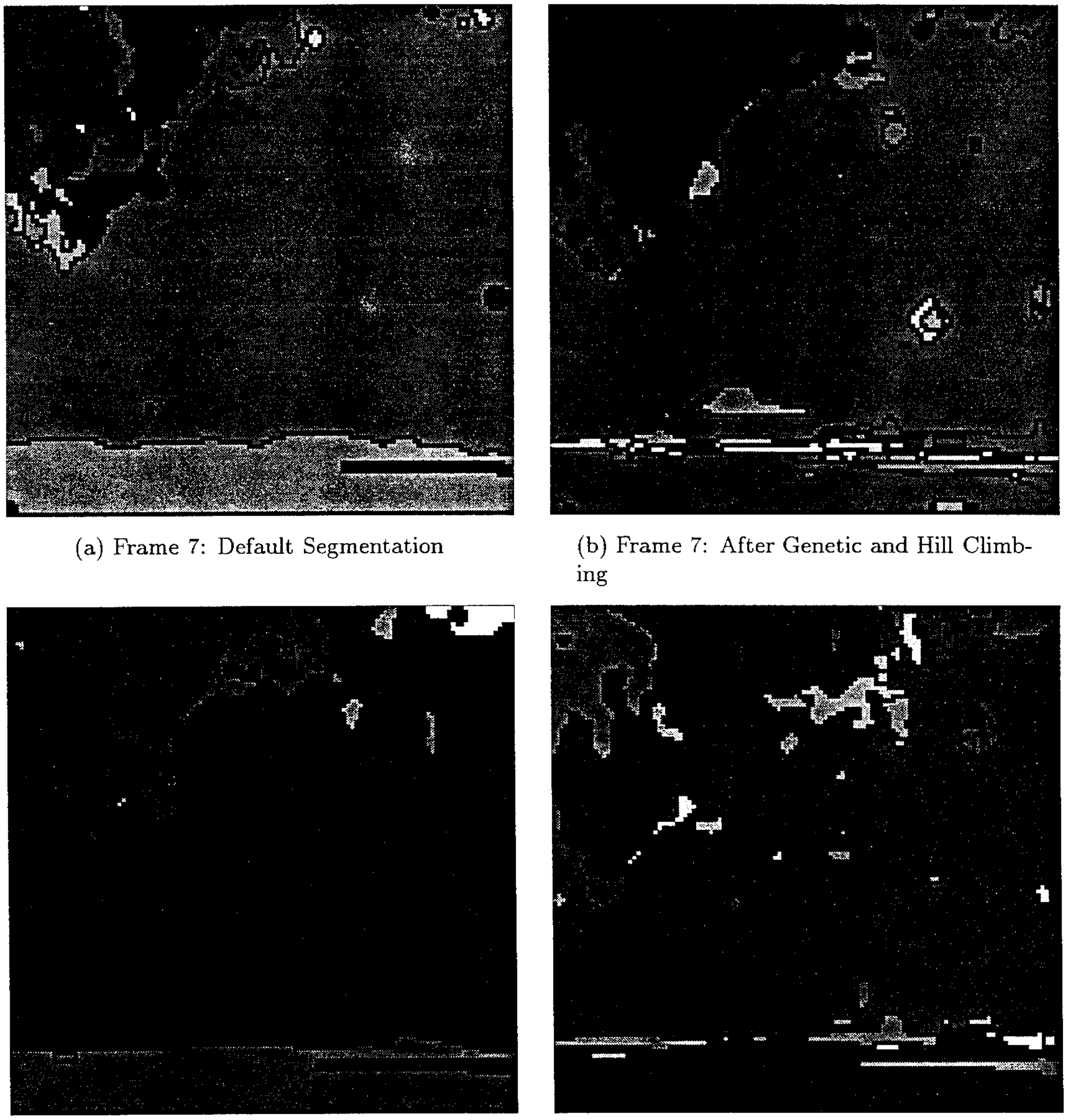
(a) Frame 2: Default Segmentation

(b) Frame 2: After Genetic and Hill Climbing

(c) Frame 3: Default Segmentation

(d) Frame 3: After Genetic and Hill Climbing

Figure 2.14: Segmentation performance comparison for frames 2 and 3: (a) Frame 2 using default parameter set, (b) Frame 2 using parameter set generated by genetic and hill climbing, (c) Frame 3 using default parameter set, (d) Frame 3 using parameter set generated by genetic and hill climbing.



(a) Frame 7: Default Segmentation

(b) Frame 7: After Genetic and Hill Climbing

(c) Frame 16: Default Segmentation

(d) Frame 16: After Genetic and Hill Climbing

Figure 2.15: Segmentation performance comparison for frames 7 and 16: (a) Frame 7 using default parameter set, (b) Frame 7 using parameter set generated by genetic and hill climbing, (c) Frame 16 using default parameter set, (d) Frame 16 using parameter set generated by genetic and hill climbing.

Chapter 3

Learnable Structural Models for Target Indexing: Hidden Markov Models, n-Grams, and Salient Sequences

3.1 Introduction

Automatic target recognition (ATR) is an image understanding problem whose goal is to find instances of "known" targets in the input sensor data. It comprises the computational processes of detection of target-like features, indexing or recognition of targets based on detected features, and verification or identification of indexed targets. Real-world ATR scenarios are characterized by multi-modal imagery, low contrast, high clutter, camouflage, partial target occlusion, and other image variabilities. Thus, the ATR problem space is represented by a number of state variables, such as target state (pose/location), sensor state (pose/location), background, environment, which account for the high dimensionality of the problem space [112]. Since a direct mapping from detected features to target models for identification is computationally expensive in such a high-dimensional space, the indexing problem is now considered as an important intermediate step in the overall recognition process. The role of indexing is one of signal-to-symbol transformation in which hypotheses about targets are framed in a bottom-up fashion, i.e., driven by detected features. The specific tasks that are involved in the indexing process are *delineation* of image regions which correspond to targets using higher-order groups of detected features, *recognition* of grouped features as specific target classes, and *localization* of recognized targets in terms of aspect, scale, and depression angle.

Current approaches to object/target model indexing lack flexibility and robustness required for ATR applications. In this work, we describe an approach to indexing that is based on utilization of weak structural models, direct table look-up, and inexact sequence matching. The weak structural models are defined as hidden Markov models (HMMs) which together with similarity-based analysis are appropriate for handling uncertainties and distortion in the imaging process; the table look up method utilizes invariant features similar to the existing approaches to indexing. HMMs, which have been effectively used in speech recognition systems are generalizations of stochastic finite automata and are amenable to learning. Besides, the look-up table (LUT) and the database for sequence matching are both constructed from examples through learning processes.

3.2 Motivation

The purpose of *indexing* is to make good guesses about an object's identity and pose (including scale) from partial evidence in a bottom-up fashion. The assumption here is that detection is generally not possible from the evidence contained in single feature cells.[1] Input to the indexing step is an unordered set (stream) of locations-of-interest (LOIs) produced by the detection module. An object hypothesis produced by the indexing module must provide sufficient information to the recognition module to initiate a goal-directed matching process. In the optimal case, this includes (1) the object category, (2) the relevant aspect, and (3) the (approximate) location in the image. In general, it will be sufficient to indicate a *set* of possible aspects or even a set of possible object categories, depending on the information available. In general, indexing is accomplished by (1) combining the information available from multiple LOIs within a certain neighborhood, (2) imposing more specific structural constraints that are suitable to narrow down the object category, and (3) by reverting to the original image data (Gabor decomposition) to obtain additional information.

The problem of indexing can be stated as "finding the needle in a haystack" (without knowing if there actually is a needle), given

1. weak local evidence,
2. spatially unrelated LOIs,
3. unknown object (target) identity, and

[1] Otherwise, indexing could be done at the same time as detection.

4. unknown target location and extent.

The general approach here is to look out for complex spatial arrangements of features in order to associate them with parts of a known model, *without* performing explicit subgraph matching with the structural model base itself. The technique we suggest here is a combination of *Hidden Markov Models* (HMMs) [84] and a variant of the *n-gram* method [111], which have both been used successfully in natural speech recognition. While HMMs provide an elegant way to model low-order (usually only first-order) dependencies between adjacent elements, n-grams represent sparse high-order (n-order) relationships that facilitate efficient indexing. Supposedly, 3-grams (*trigrams*) will be sufficiently powerful for indexing.

The advantage of using HMM and n-gram techniques is their use of a discrete alphabet of symbols, which allows the use of a probabilistic inference scheme, and the existence of fast evaluation algorithms. The crucial point with symbolic methods is the need for a "pixels-to-symbols" transition before they can be applied. *Hidden* Markov models (in contrast to conventional Markov models) support this transition well, because they explicitly handle the problem of uncertainty of pixel-to-symbol association.

The main obstacle to a direct application of the above methods is that we have to deal with 2-D configurations instead of the 1-D sequences in speech recognition. One solution to this could be a 2-D random walk over neighboring features, (softly) biased by certain selection criteria that can be learned from experience.

3.3 The Domain of Learning

Figure 3.1 describes our overall learning-based approach to indexing. The input to the system consists of various structural primitives which have been obtained by low-order grouping such as perceptual grouping of edge-based features, e.g., smooth, elongated curves, or region-based features, e.g., blobs. To utilize the HMM paradigm for evidence accumulation, we adopted a *discrete symbol* HMM. Consequently, the input stream of continuous observables need to be discretized and subsequently sequentialized. Once the discrete observation sequences have been obtained, these can be used as input to an HMM-based process for classification. There exist efficient real-time algorithms to decompose the input sequence into meaningful state sequences of an HMM (discussed below). For example, the individual HMMs can be associated with target classes and states with subparts of a target. Thus, uncovering of the states will result in segmentation of the input sequence into subparts, i.e., subpart decomposition. Also, HMM outputs the best model or the best indexed target class (see Figure 3.1). By associating observation symbols with states, one obtains higher-order grouping. This segmented sequence of observations can be further utilized to determine

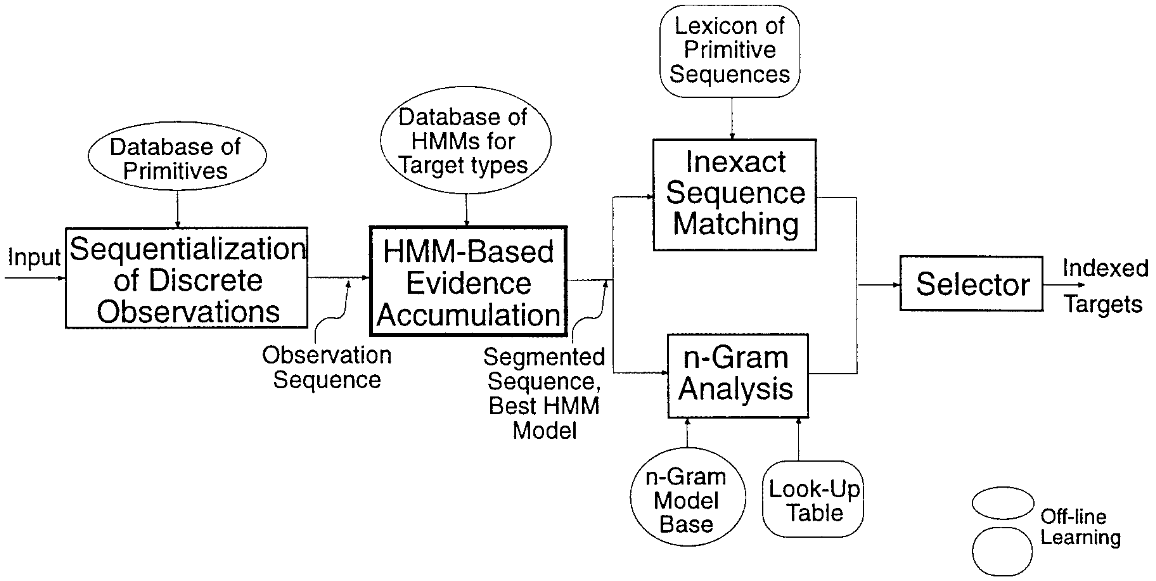

Lexicon of Primitive Sequences
Database of Primitives
Database of HMMs for Target types
Input
Sequentialization of Discrete Observations
HMM-Based Evidence Accumulation
Inexact Sequence Matching
Selector
Indexed Targets
n-Gram Analysis
Observation Sequence
Segmented Sequence, Best HMM Model
n-Gram Model Base
Look-Up Table
Off-line Learning

Figure 3.1: A schematic of the learning-based approach to target indexing.

the class of target apart from what HMM outputs. In our approach, we have two alternatives. One is the fast direct table look-up based on n-gram analysis of the segmented state sequence. The other one is slower inexact matching using a lexicon of primitive sequences. The selector at the output has to its disposition indexed targets generated by HMM, n-gram, and inexact matching.

3.3.1 Hidden Markov models

A Hidden Markov model is a stochastic signal model and is an extension of the theory of Markov chains. According to this theory, at any given instant, a stochastic system can be in one of a number of distinct states. At regularly spaced discrete times, the system undergoes a state change with a certain probability. HMMs have been widely used with considerable success to classify signals in speech and text recognition systems [54, 58, 83, 84]. In fact, it is a good choice for decision making in situations involving *sequences* of observations.

Figure 3.2 illustrates the phenomenon underlying the HMM framework. The Markov chain of states is a probabilistic description of a stochastic system at any given instant. In the case of HMM, this stochastic process is hidden and can only be observed through another

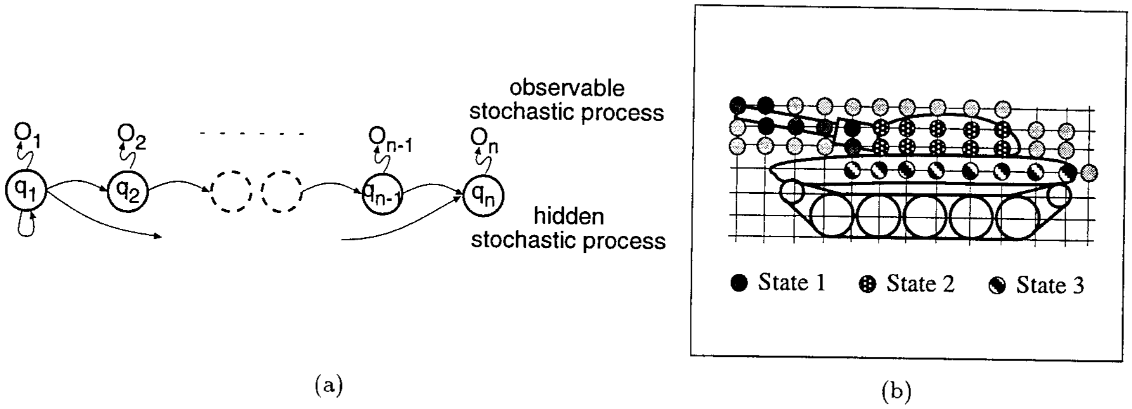


Figure 3.2: Discrete symbol hidden Markov model: (a) HMM as a doubly stochastic process, (b) an example illustrating HMM where the observations of the same state are indicated using the same symbol.

stochastic process which produces the sequence of observations as seen in Figure 3.2(a). Thus, an HMM is a doubly stochastic process. To illustrate these basic ideas, consider an image consisting of a tank as shown in Figure 3.2(b). Suppose, the tank is represented using an HMM whose different states correspond to different subparts of the tank. Let the input observations consist of a set of filter responses which are obtained at the lattice points of the grid shown in Figure 3.2(b). Thus, the different subparts are only observed through these filter responses. For example, State 1 corresponds to the gun, State 2 corresponds to the turret, and State 3 corresponds to the body.

There are several elements in the definition of a hidden Markov model [83, 84]:

- number of states (N) – these are the distinct states $S = \{S_1, S_2, \cdots, S_N\}$ which the system can be in,
- number of symbols (M) – these are the distinct symbols $V = \{v_1, v_2, \cdots, v_M\}$ which are observed in any state,
- state transition probability (A) – the probability distribution set is $A = \{a_{ij}\}$, where $a_{ij} = Pr[q_{t+1} = S_j | q_t = S_i]$, $1 \leq i, j \leq N$,
- observation symbol probability (B) – the probability distribution set is $B = \{b_i(k)\}$, where $b_i(k) = Pr[O_t = v_k | q_t = S_i]$, $1 \leq i \leq N$ and $1 \leq k \leq M$, and

- initial state probability (π) – the probability distribution set is $\pi = \{\pi_i\}$, where $\pi_i = Pr[q_1 = S_i]$, $1 \leq i \leq N$.

A complete specification of an HMM requires specification of N and M, specification of observation symbols, and specification of the three probability measures collectively denoted as $\lambda = \{A, B, \pi\}$.

There are three modes in which an HMM can be operated [83, 84]:

- *Training* – It involves adjusting model parameters λ to maximize the probability of observations, i.e., maximize $Pr[O|\lambda]$.
- *Classification* – Assuming that each λ is associated with a class ω, it involves selecting that λ which optimally explains the observation sequence and, therefore, identifies the class, i.e., $\omega = \text{argmax}_i Pr[O|\lambda_i]$. Classification also allows uncovering of the optimal state sequence.
- *Generation* – It allows generation of an observation sequence O (of usually a specified length), given N, M, and λ.

In this work, we focus on the first two modes, i.e., *training and classification.*

The important difference between application of HMM to speech or text processing and 2-D image analysis is finding *sequences* of observations. In the former case, there is a natural order, temporal or spatial, in the observations. However, in the latter case this is seldom the situation, except under restricted imaging conditions, such as observations are ordered along a row or column. Thus, the application of HMM to general image analysis requires a solution to the problem of *observation sequentializing.* This problem which is essentially the *where to look next* problem does not have a general solution and in case of humans has been demonstrated to be varying with time and from person to person for a given input pattern. Consequently, computational approaches have proposed criterion which are task-specific. In our application context, these sequences of observations (associated with different targets) are to be obtained by ordering the observations in some meaningful way which is independent of the target present in the input image. One idea to find sequences of meaningful observations is to locate *salient* structures in images. The approach of Sha'ashua and Ullman [100] finds sequential arrangements of salient image locations represented by perceptually long and smooth curves. Figure 3.3 shows a grey-level image and salient structures detected in the corresponding intensity edge image. In this work, we investigate a *more general* framework for the ordering (sequentializing) problem that is independent of the type of observations, i.e., applicable to both edge- and region-based observations. We also discuss the sequentializing problem at the object level when the salient structures are

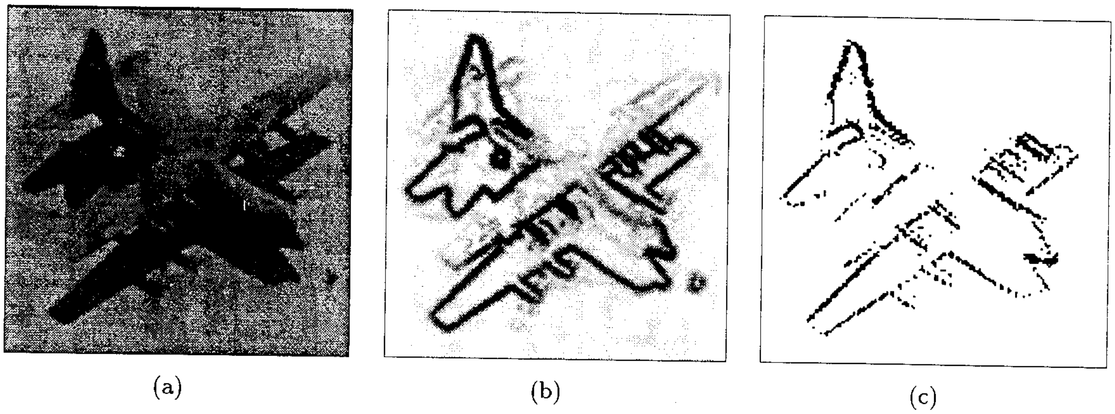


Figure 3.3: Extraction of salient structures: (a) original image, (b) edge image, (c) high saliency map using (b).

given (since salient structure extraction addresses the sequentializing problem at the pixel level).

Additionally, it is to be noted that in order to use discrete symbol HMM, the observations of the sequence must be discretized with respect to their intrinsic values such as length, orientation, location of line features and elongation, orientation, location of blob features.

Observation Sequentializing as a Markov Decision Process

In the general framework, the ordering problem is considered to be a *multi-stage decision process* which is modeled as a Markov chain. Associated with each state, S_i, of the Markov system is a discrete, finite set of actions, $A(i)$, and a similar set of observations, $\Theta(i)$. Based on the observation, $\theta \in \Theta(i)$, at a given instant, the controller selects an action, $\alpha \in A(i)$, which transforms the system to a new state at the next instant. The important difference between a Markov decision process and (hidden) Markov model is that no action is allowed to influence the state transitions in the latter. In a first-order Markov model, the probability of the transition from the current state, $s(t) = S_i$, to the next state, $s(t+1) = S_j$, is expressed as

$$p_{ij} = Pr\{s(t+1) = S_j \mid s(t) = S_i, \theta(t) = \theta, \alpha(t) = \alpha\}, \quad S_i, S_j \in S, \tag{3.1}$$

where S is the finite set of discrete states of the system. For observation ordering, each input feature (an observation) is assumed to be associated with a state S_i, where the state-based observation θ corresponds to certain measurable properties of the selected feature, such as length, orientation, color, size, etc. The typical actions at each state may involve searching in a *quantized direction* ϕ over a *quantized distance* d in the image plane. Thus, the problem of feature ordering may be formulated as one of "finding a state *sequence* such that the corresponding feature (observation) sequence is *significant* in some chosen sense".

The extraction of a sequence of significant states depends on the appropriateness of the selected actions. According to the theory of Markov decision tasks, there exists at least one policy, i.e., mapping from states to actions, which is *optimal.* The criterion for optimality is usually defined in terms of the *expected discounted reward-to-go.* Let the reward for entering state S_i as a result of selecting the action α be denoted by $r_i(\alpha)$. Then, the expected discounted reward-to-go in S_i, $F_i(\alpha)$, is an infinite sum of the expected future rewards, each of which is weighted by a decreasing (temporal) discounting factor:

$$F_i(\alpha) = r_i(\alpha) + \sum_t \gamma(t) R(t), \tag{3.2}$$

where $\gamma(t)$ is the discounting factor and $R(t)$ is the expected reward in t time steps. Consequently, Equation 3.2 can be written as a recursive function of the successor states of S_i:

$$F_i(\alpha) = r_i(\alpha) + \gamma \sum_{j \in \mathbf{succs}(i,\alpha)} p_{ij}(\alpha) F_j(\beta), \tag{3.3}$$

where $\mathbf{succs}(i, \alpha)$ is the set of successor states of S_i as a result of the action α, $\beta \in A(j)$, and γ is the discounting factor in the next time step. Now, the goal of the controller is to select an optimal policy $\alpha = \alpha_{opt}$ which maximizes the expected discounted reward-to-go $F_i(\alpha)$. This is obtained as the root of the equation $\nabla F_i(\alpha) = 0$ when $F(\alpha)$ is a continuous, convex function over the space of α. When the transition probabilities p_{ij}'s are known, the expected discounted reward function is completely specified and the optimal solution is obtained using such well-known methods as Dynamic Programming (DP). However, when these probabilities are unknown, the optimal solution is incrementally *learned* on the basis of the observations – the series of actions, state transitions, and rewards.

As noted earlier, the use of discrete symbol HMM requires quantization of the measured values of observations. For example, a line feature would need its orientation or length value discretized. In speech, the corresponding problem is known as *codebook design* for mapping continuous observation vector into a discrete codebook index. One approach to represent a quantized observation vector is to obtain the probability density function (pdf) for the corresponding cell of the observation space. In this approach, a cell is represented as a family of overlapping Gaussian pdf's. This probabilistic view is well suited for incomplete data. Re-estimating the parameters of the mixture pdf's is thus an unsupervised learning problem.

Learning methods

Learning Control of Observation Sequentialization: The field of *reinforcement learning* [10, 38] is concerned with the study of learning control of Markov decision processes. This learning paradigm essentially finds a solution action $\alpha = \alpha_{opt}$ (refer to Equation 3.3), given the current state, a set of possible actions, and past experience of success and failure.

A sequence of observations captures a *global context* in the sense that the ensemble is structurally significant and not the individual components. Thus, the significance measure (saliency according to [100]) of the local observations must propagate along the sequence to result in its global significance. The reinforcement learning paradigm is particularly suitable for this class of problems since structural saliency requires measures that have a global extent and reinforcement learning is applicable to control problems involving temporally extended behavior. To support propagation of local evidence in a global fashion, an appropriate architecture for state space must be chosen. In our approach, the image is represented as a grid of processing elements (PEs) each having a fixed neighborhood. A PE is completely connected to its neighborhood PEs which in turn contribute to its significance measure. This architecture is similar to the saliency network model introduced in [100]. Reinforcement learning methods have been demonstrated most successfully for connectionist networks [10]. We assume that the input observations are either edge- or region-based. Thus, a PE is associated with the presence or absence of an edge or region pixel. An ordered set of observations is a *sequence* of PEs which is optimal according to a certain saliency measure.

To cast the network training for salient observation sequence extraction as a reinforcement learning problem, we adopt the following mappings: a PE corresponds to a state, preference for a particular type of curve or linear arrangement of blobs corresponds to an action, saliency value at a PE corresponds to expected discounted reward-to-go. Thus, in the process of finding the optimal action, the learning process biases the network (for a chosen saliency measure) towards detecting certain feature groups based on the training examples. For any given state, i.e., a PE, the set of its successor states consists of all the PEs in its neighborhood. The selection of a particular action in a state, such as selection of a curve whose tangents are near-horizontal, causes some of the successor states to be more preferred than others. Since the action value is usually quantized, such as curves of tangential slopes less than 10° preferred, the selection of the next state is probabilistic.

The two critical issues in the application of reinforcement learning are *how to explore the state space* and *how to generate the reward.* The importance of the first issue lies in observation that for a high-dimensional state space an exhaustive exploration strategy would be computationally prohibitive. Since the state space is of a high-dimensionality in our approach (number of states equal to the total number of pixels), a non-uniform partitioning of the space is required for efficient exploration. One way to obtain such a partitioning is to use a kd-tree representation [93] for the image. The second issue or reward generation is important because it is related to effectiveness of learning and the learning rate. We employ a scalar feedback value, r, to penalize selection of a state which is not associated with any edge or region pixel; $r = 1$ if state is associated with an edge/region pixel, $r = 0$ otherwise. The generation of the reward can occur at every step or it can be delayed over several actions steps. In our current approach, the reward is generated at every step so that the network converges at a faster rate to a reasonable solution.

There exists a number of approaches to reinforcement learning in the literature [10]. Our initial attempt aims to utilize these for our problem instead of developing a new one (a future issue). We plan to investigate two of the most recent approaches, Q-learning [107] and Prioritized Sweeping [70]. Both of these approaches are asynchronous dynamic programming (DP) techniques, that is they seek an optimal policy which specifies an action such that the expected discounted reward (refer to Eq. (3.3) is maximized:

$$\bar{F}_i(\alpha) = \max_{\alpha \in \mathbf{actions}(i)} \left[\hat{r}_i(\alpha) + \gamma \sum_{j \in \mathbf{succs}(i,\alpha)} \hat{p}_{ij}(\alpha) \bar{F}_j(\beta) \right]. \tag{3.4}$$

Here, **actions**(i) is the set of all possible actions in state i, $\hat{p}_{ij}(\alpha)$ is the estimated probability of the transition from state S_i to state S_j given that the action α has been applied, and $\hat{r}_i(\alpha)$ is the estimated reward so far from all previous applications of action α in state S_i.

Q-learning is a model-free approach in that it directly learns the optimal policy without building a world model. It uses a local greedy strategy specified by Eq. (3.4) to select the locally optimal action. The new estimated discounted reward-to-go is combined with old estimate using a weighted sum:

$$F_i^{new}(\alpha) = (1-c)F_i^{old}(\alpha) + c\bar{F}_i(\alpha), \tag{3.5}$$

where c is the learning rate.

Prioritized Sweeping , on the other hand, extensively relies on learning a world model from which a control rule is developed. It concentrates its computational effort on the most "interesting" parts of the system. These are identified with those states for which maximum change in expected discounted reward-to-go occurs. After each real-world observation $S_i \rightarrow S_j$, the transition probability $\hat{p}_{ij}$ is updated along with the probabilities of transition to all other previously observed successors of S_i. Thus, Prioritized Sweeping is much more memory intensive than Q-learning since the former needs to build the world model. The expected discounted reward-to-go is updated for every state S_i and all the predecessors of the interesting states. The order of update for the predecessors is determined by a priority value of each which is based on the amount of change in the reward value of the successor and the transition probability between the predecessor and the successor.

Learning Observation Quantization: Earlier, we have described the problem of observation quantization as one of estimating the parameters of a mixture Gaussian pdf's representing each quantized cell. A well-known maximum likelihood estimation technique is the EM algorithm [32]. This algorithm, which is an unsupervised learning method, works with an unlabeled data set assumed to have been derived from an unobservable category (similar to the notion of HMM). The purpose of EM algorithm is to maximize the log-likelihood from incomplete data, i.e., observable data, by iteratively maximizing the expectation of log-likelihood from complete data, i.e., observable and unobservable data.

Learning HMMs for Target Classes: Each target class, such as tank, truck, APC, aircraft, is represented by an individual HMM and the objective is to build these HMMs from training data. The input data consist of structural primitives whose feature values are quantized. The vocabularies of structural primitives may be identical for the observations and the hidden state sequence. Usually, the vocabulary of observations is larger than that of the state sequence. The goal of this training phase is to determine the various elements of an HMM corresponding to a target class.

Usually, the parameters N and M, the numbers of discrete symbols/state and the number of states, are selected prior to the training process. For example, the number of states in HMM of a target class may well be the number of distinct components of structural decompositions, such as gun, turret, body, and wheel of a tank. This selection of states is useful when the target image can be segmented into meaningful subcomponents, e.g., based on extracted contours. On the other hand, if such segmentation is not possible, then it is more prudent to select identical vocabularies for observation and state sequences. It is to be noted that an observation corresponding to a state is usually vector-valued. The remaining parameters of HMM to be learned are the various probabilities.

A) Given input sequence, learn λ – Following the general framework of observation sequentialization, it is assumed that the input *sequence* of observations is already extracted. Subsequently, the learning of HMM models involves the application of an iterative algorithm, the Baum-Welch algorithm [83, 84]. This algorithm is used to estimate the probabilities from frequency of observations. It is in sense a maximum likelihood estimation as the EM algorithm. Consequently, the algorithm progresses as selecting the initial model λ and reestimating another model $\bar{\lambda}$ such that either λ defines a critical point of the likelihood function, in which case $\bar{\lambda} = \lambda$, or $\bar{\lambda}$ is more likely in the sense that $Pr[O|\bar{\lambda}] > Pr[O|\lambda]$. Therefore, by iteratively replacing λ with $\bar{\lambda}$, the probability of O being observed is increased until some limiting point is reached. The result is the estimated model.

B) Find input sequence, learn λ – This situation is encountered when the general framework of observation sequentialization is not adopted, e.g., input consists of salient structures of curves such as those derived from Figure 3.3(c). In this case, the problem of finding the input observation sequence is linked with deriving the Markov models. We assume that the input consists of salient curves, each of which can be modeled, e.g., using B-spline or polygonal approximation, as an autoregressive process, etc. The parameters of the model representation are used to form the feature vector which is then subjected to the quantization process described above. One or more these quantized feature vectors are the observations associated with a state of an HMM. The model learning problem in this case is equivalent to

$$\max_{i,j} Pr[O_i, S_{opt}|\lambda_j],$$

where O_i is the ith input sequence, S_{opt} is the optimum state sequence, and $Pr[.]$ is known as the state-optimized likelihood function. For a given λ, an efficient way to find $Pr[O, S_{opt}|\lambda]$ is the well known Viterbi algorithm [24, 83, 84].

During the training phase, all the input feature vectors are clustered, e.g., using k-means algorithm. The cluster centers are then chosen as the states of an HMM. Each feature vector in a cluster is assigned the state which is the cluster center. The transition probabilities and the initial state probabilities are computed based on training sequences. Viterbi algorithm is then used to trace the optimal state sequence S_{opt} for each selected sequence O_i of the feature vectors. A vector is reassigned a state if its original state assignment is different from the tracing result and the processes of model learning and state tracing are repeated.

Implementation Issues: There are several issues of practical importance from the implementation point of view. These include scaling, multiple observation sequences, initial parameter estimates, missing data, and choice of model size and type [83]. Scaling of probabilities is important since small probability values often exceed the precision range of any machine. Multiple observation sequences from the same source are necessary to have sufficient data for reliable estimation of model parameters. The importance of selecting an appropriate initial parameter set (the initial model) is well-known for any gradient technique. A solution to this problem is to administer a supervised training procedure with manual segmentation of observations into states. Alternately, an unsupervised training using k-means segmentation with clustering can be followed. To overcome the problem of insufficient data for reliable model estimation, the size of the training data can be increased, the size of the model can be decreased, one set of parameters can be interpolated using another set from a model receiving sufficient training data, or extra constraints can be added so that every parameter value satisfies them. The choice of the model, i.e., size and type (e.g., left-right or ergodic), is usually application dependent.

The order of the model, i.e., the number of intermediate states skipped, determines the ability of the classification process to handle distortion; higher the order, greater is this ability. If the observations are made scale- and aspect-specific with respect to the targets, then scale- and aspect-dependent HMMs are obtained for each target class.

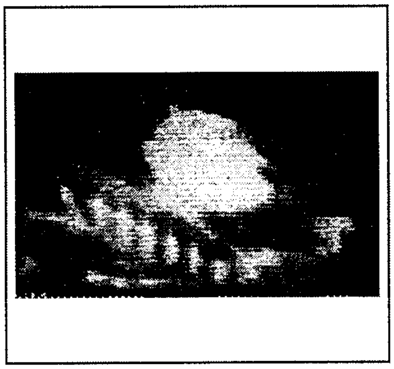
Performance Evaluation

The performance of HMMs for target indexing can be evaluated in terms of several criteria: classification accuracy, model trainability, input pattern distortion.

An efficient learning method is capable of reproducing the same high performance learning behavior under similar situations during training and testing. Using HMMs, the learning behavior is evaluated in terms of the number of correct classification of the input patterns. The classification accuracy depends on the choice of the model parameters. For example, increasing the number of states usually leads to an improvement in the accuracy. Usually, the Viterbi algorithm is applied for classification purposes. As noted earlier, the Viterbi algorithm yields both the best matching HMM and the state sequence. During training, the error between the expected state sequence (determined by the observation sequence) and the extracted (via Viterbi algorithm) sequence is used to reestimate the parameters of the HMMs until the error is small. It is expected that such reestimation procedures would lead to improved classification results. Both training and test patterns should be representative of the variety of scenarios in which a particular target is expected to be found in order to ensure a robust classification performance.

The primary issue related to the trainability of the HMMs is the amount of data. Larger number of states of HMMs implies greater amount of training data to estimate the model parameters more accurately. Although, we have discussed some alternate solutions to overcome the data insufficiency problem earlier, a trade-off between classification accuracy and trainability is always required while selecting the size of an HMM. Since the training data have to cover various scenarios, selection of the HMM parameters become even more important so as to keep the training data set of manageable size.

The performance of the HMMs to accurately classify the input observations depends on their ability to generalize from the training data. To exhibit robust classification performance, the HMMs must be able to handle significant pattern distortion caused by noise, occlusion, and viewpoint location. Thus, the training data need to include all such excep-



(a)

blob	0.95
elong	0.85
period	0.35
blob-elong	0.70
blob-period	0.25
elong-period	0.15
elong-blob-period	0.05

(b)

Figure 3.4: n-Grams of patterns typically encountered in ATR imagery: (a) FLIR image of a tank, (b) uni-, bi-, and tri-grams of patterns, such as blob, elongated shape, periodic structure, and probabilities of occurrences in images such as (a).

tions to ideal data.

3.3.2 n-Grams

Like HMMs, n-grams have also been very successfully used in natural language and speech processing and information retrieval [104, 111]. An n-gram is a string of n consecutive symbols from the same alphabet. The elements of an n-gram model, γ_n, are,

- n-grams for a selected value of n,
- the corresponding probabilities.

The probabilities are obtained by gathering statistics from observations. In Figure 3.4(b), we give examples of uni-, bi-, and tri-grams of patterns which may be typically encountered in ATR imagery such as Figure 3.4(a). The probabilities indicated in Figure 3.4(b) are merely to illustrate the n-gram model.

An n-gram model can be utilized in two ways:

- *Verification* — Given a model γ_n, it involves verification that an element $\epsilon \in \gamma_n$ matches the observation sequence O. Consequently, the element is identified, i.e., $\epsilon = \text{argmax}_i Pr[O|\gamma_{ni}]$.
- *Generation* — It allows generation of an observation sequence O (of usually a specified length), given γ_n.

In this work, we focus on the first mode, i.e., verification.

Learning method

The n-gram analysis is designed to extract indices of targets from a lookup-table (LUT) based on the detected substructures, i.e., the n-grams. Thus, the goal of learning is to design the LUT based on input data and expected output during training. The choice of n is important since a large value of n may make the n-gram somewhat unique or an invariant feature, but at the same time such substructures are difficult to detect. A reasonable choice of n is $3 \leq n \leq 5$.

The learning task can be formulated as, given a sequence of discrete symbols and the corresponding labeled target, determine the n-grams to be used for indexing that target. Clearly, this is a problem of supervised learning. The n-grams can be directly used to index the LUT or these can be hash-coded if their direct use increases the size of the LUT considerably. Here, we assume that the n-grams index into a hash table. Thus, the learning process must discover the partitioning of the space of n-grams such that each partition contains a pointer to a substructure of a target model. This is a well-known problem in pattern recognition, a solution to which may be obtained using decision trees. An advantage of using decision trees is that the classification rules provide a clear explanation of the classification process. However, there exists no unique decision tree for ascertaining the partitions needed for a hash table.

Decision trees are commonly used to represent domain knowledge [81]. Each nonterminal node in a decision tree specifies some attribute and each branch specifies the alternative values. A decision tree classifies a new instance by repeatedly sorting downwards while selecting the most discriminating attributes and exploring the branches associated with each of the attribute values until a terminal node, specifying a class name (a target in our case), is reached. The attributes of a decision tree thus correspond to those of the hash table, i.e., the axes of the table. The creation of the hash table occurs via the induction of a decision tree. In other words, the partitioning along the various axes of a hash table correspond to the tests at the different nodes of a decision tree.

In order for a hash table to be optimal, the classification error using the hash table must be minimal. In general, finding the optimal table requires exhaustive search over all possible partitions of the n-gram space. Thus, an indirect approach to constructing the hash table consists of constructing an approximate decision tree initially. Next, the decision tree is converted into an equivalent hash table. The decision thresholds applied at each node of the tree partitions the attribute space of the hash table. This process may create additional partitions which may not be present in the direct construction of the hash table. An important consideration for decision tree construction is over-specialization or "overfitting" of the input examples due to reduction in learning bias. This overfitting increases the variance or noise sensitivity of the learning process. A related issue is "underfitting" or overgeneralization. Since the goal is to classify as many input samples as possible by reducing the overall classification error, a decision tree has to be appropriately pruned during the learning process.

Performance evaluation

The performance of n-gram method for target indexing is evaluated in terms of error rate. The error rate is measured in terms of hits and misses, and false alarms. It is related to the number of terminal nodes of a decision tree or the attributes of the hash table, and decreases with the increase in this number. However, for practical purposes the choice of n from which these attributes are derived is small as noted earlier. Since pruning of a decision tree is a necessity to allow generalization capability, the indexing performance is influenced by the selection of the pruning method, e.g., pruning based on partitioning samples into training and test sets, pruning by resampling input, pruning by identifying the weakest link in the tree.

Another relevant performance measure for any supervised learning is in example complexity or the number of training examples required for a satisfactory performance. If the number of training samples is small, then the growth of the decision tree will be restricted and subsequent pruning will lead to poor generalization capability. The phenomenon of missing or spurious data causes inadequate coverage of the space of learning experiences. In such case, the number of training examples have to be increased.

3.3.3 Inexact sequence matching

The inexact sequence matching is concerned with approximate matching of two sequences, an input and a stored sequence, containing structural primitives and their spatial relations. The inexactness or approximateness of the match is to allow for occasional large distortions in the input patterns, such as due to moderate to large occlusion, high clutter, deep hide, etc. As noted earlier, this approximate matching for indexing is more of a backup process when more precise matching methods based on n-gram and HMM would fail. The complexity of such techniques depends on the matching strategy and the method of storing sequences in the lexicon (see Figure 3.2).

The notion of "inexact" may be interpreted as one of *equivalence* – the two matching sequences are essentially the same except for small differences – or *similar* – the matching sequences could actually be different but appear similar due to the large variability. If the threshold of acceptance is set too high to limit *similar* sequences, then it is expected that many *equivalent* sequences would be missed out. A lowering of the threshold, on the other hand, would allow more similar sequences causing false alarm. One way to locate an equivalent pair is to define a *canonical form* which is the representative of the equivalence class (using the mathematical definition of equivalence) to which the pair may belong [49]. For example, in text processing two equivalent sequences are alternate spellings of the same word and the canonical form is created by transforming these sequences into some standard spellings. In our case, finding the canonical form is seeking a generalization of the equivalent sequences. To process similar sequences, a measure of similarity based on a similarity or a difference metric is required. Understandably, the similarity metric is for sequences belonging to the same equivalence class while the difference metric is suitable for sequences belonging to different equivalence classes and should therefore be based on the corresponding canonical forms. In either case, the chosen metric must model the source of variation correctly, otherwise ascertaining an inexact match in a large lexicon of sequences is difficult. Some of the successful methods [49] in speech and text processing are based on dynamic programming, which attempts to find the shortest path in a graph whose nodes are labeled by pairs of elements drawn from the two sequences, and computing probabilities of joint occurrences of element pairs in the two sequences to obtain the joint event of the matching sequences.

Learning method

Selecting the size of the lexicon is an important problem for satisfactory performance. For a large size lexicon, the number of undetected errors tend to be higher. This is because there would always be some sequence in the lexicon which would approximately match an input sequence. Consequently, the choice of correct context becomes very important in matching sequences using a large lexicon. Another problem with large lexicon is that the search time is proportionately longer. A better approach is to employ a learning method to grow the lexicon so that it is tailored to the specific application, e.g., target recognition using FLIR images, and yet its size is manageable.

An input sequence consists of alternating structural primitives and spatial relations (binary) between adjacent primitives in the sequence. Additionally, we assume that each target class is represented by a set of sequences, where the different sequences might capture the differences in the appearances of the specific members of the target class, possibly under varying imaging conditions. Matching of two sequences can then be formulated as a nearest-neighbor classification problem. In this paradigm, a learning approach is required for the following tasks: (a) addition of a new sequence to the lexicon corresponding to an existing/new target class, (b) deletion of an old sequence from the lexicon, (c) generalization of two matching sequences, (d) refinement of an existing sequence.

The similarity between learning a class of sequences in our case and concept learning in AI motivates the use of a noise-tolerant instance based learning (NT-IBL) algorithm [56]. This framework extends the nearest neighbor (NN) algorithm by generating classification predictions using only specific instances without maintaining the abstractions derived from specific instances. IBL offers the advantages of simple representations for learnable entity (i.e., sequence) descriptions, low incremental learning costs, small storage requirements (e.g., compared to NN), ability to learn continuous functions and non-linearly separable categories. The NT-IBL uses significance tests to distinguish noisy instances in the training examples. It seeks evidence that saved instances are significantly good classifiers before it allows them to be used for subsequent classification tasks.

The simplest form of IBL is a growth algorithm. Given C as the concept to be learned and T as the training set, IBL initializes C to the set containing the first element in T. For each subsequent element in T, the algorithm finds the nearest neighbor in C to the current element in T. If the current element is correctly classified, then it is discarded, otherwise it is appended to C. In the NT-IBL, the classification records of all instances in C that are at least as similar to the current element in T as the nearest neighbor in C are updated. Also, the instances in C which appear to be noisy after the application of the significance test are discarded.

Both NN and IBL work with vectors of features that are of the same length. In contrast, the sequences in our approach like their counterparts in speech and text processing are of variable length. Thus, to use IBL, these sequences have to be made of a predefined fixed length by incorporating null primitives and spatial relations during run-time. To evaluate the similarity between an input vector and an instance in C, IBL uses the euclidean distance metric. Since the sequences consist of primitives and spatial relations, appropriate distance metrics have to be used for the primitives and the relations separately. Comparison of primitives is based on type equivalence, i.e., line to line, blob to blob. For same type of primitives, e.g., ith primitive of the input sequence T and jth primitive of an instance sequence S are of type P (say, ellipse), the distance measure $d(T_i, S_j)$ is based on the cumulative normalized distance between attribute values, such as the Minkowski metric of order n:

$$d(T_i, S_j) = \left[\sum_{k=1}^{m} \left| \frac{T_{ik} - S_{jk}}{V_{Pk}} \right|^n \right]^{1/n},$$

where m is the number of attributes of the primitive type P and V_{Pk} is a normalizing value of P corresponding to the kth attribute. When the relations are expressed in qualitative terms such as left, right, top, bottom, the matching algorithm seeks verification in such terms and the results are expressed as either 1 (match) or 0 (no match). On the other hand, when the spatial relations are expressed quantitatively such as in terms of angles and distances Minkowski metric can again be used.

To classify an input element, IBL uses a single instance from C, viz., the nearest neighbor. In contrast, our approach accumulates matching scores from all sequences belonging to a concept or a target class. Thus, the score for matching an input sequence T to the instance sequences $S^i \in \mathcal{S}$ of the target class C_k is

$$W_k = \sum_{S^i \in \mathcal{S}} \Phi(d(T, S^i)),$$

where the function Φ can be chosen as an error function such as a Gaussian or a Lorentzian to increase the support of similar instance sequences and reduce that of dissimilar ones. Classification is based on the highest accumulated score, i.e., C^* is the assigned class for T if

$$C^* = \text{argmax}_k(W_k), \qquad k = 1, ..., n_c,$$

where n_c is the number of target classes. In order to keep the size of each class of sequences manageable, instance sequences that are nearly equivalent are replaced by their canonical sequence derived by generalizing upon the attribute values of primitives and spatial relations (quantitative). A book-keeping procedure deletes sequences from target classes whose support classifications fail to agree with their respective target classes.

Performance evaluation

The performance of the IBL-based sequence matching can be evaluated using the measures used for any supervised learning method. These are the classification accuracy, example complexity, and noise sensitivity. The classification rate is expected to decrease with increase in lexicon size and sequence length. The example complexity requires a little different perspective than usual supervised learning approaches. This is because IBL being an incremental algorithm can utilize new information in added sequences to continue the training process. On the other hand, being an incremental process, the performance of IBL may be affected by the order in which the training examples are presented. Thus, to evaluate the performance based on the number of training examples, is has to be ensured that the test examples do not alter the learned sequence classes and also the order of the training examples need to be accounted for. The performance may be further evaluated by providing training examples corrupted with various amounts of noise.

3.4 Conclusions

Our approach has been motivated by the success of the use of weak structural models in speech and text recognition. However, in making these models effective for object recognition one faces the same level of difficulty as extending 2-D pattern matching techniques to 3-D. We plan on extending the HMM framework if it is necessary to ensure a satisfactory performance of the system or to enhance it. Subsequently, we shall be working on the other learning aspects of the indexing system, i.e., n-gram and inexact sequence matching.

Chapter 4

Signal to Symbol Conversion for Structural Object Recognition Using Hidden Markov Models

Structural recognition methods are usually based on the availability of structural primitives and the assumption that these elements can be extracted from the image data. However, in many practical situations it is difficult to extract these primitives with sufficient reliability. Regardless of what these primitives are, their extraction normally requires early decision-making at a low level and without consideration of the spatial context. Our approach attempts to avoid early segmentation by using a context-dependent classification scheme based on Hidden Markov Models (HMMs). They offer several features that make them attractive for pattern detection and matching under distortion, partial occlusion, and noise. However, due to their fundamentally one-dimensional nature, the application of HMMs to images remains a challenging and largely unsolved problem. In this chapter, we have adapted HMMs for 2-D for shape indexing and recognition. Initial results from experiments with the HMM-based indexing mechanism are shown.

4.1 Introduction

In this chapter, we study the problem of two-dimensional object recognition under the general assumption that three-dimensional objects can be represented by a finite set of 2-D aspects. However, since each aspect must necessarily cover a certain range of viewing angles, viewing distances, and possible articulations, we require that the underlying recognition process be able to handle a sufficient amount of variation between the image and the corresponding prototype aspect. The critical problems for 2-D recognition are (a) tolerance against changes in lighting conditions and contrast, (b) the handling of geometrical deviations, such as rotations, scale changes, and moderate object deformations, (c) tolerance against changes in background, and (d) partial object occlusion. A central premise is that the *shape* of an object is its most discriminating property, one that is mostly unaffected by lighting conditions and sensor parameters. The shape of an object can be defined in many different ways, for example as the occluding contour, a set of straight line segments, or a collection of bright blobs. Here we mean by shape, in a very general sense, the spatial arrangement of structural information, which typically includes boundary information as well as information about the *interior* structure of an object. The need to perform recognition under partial object occlusion requires that objects can be identified from their parts or at least a subset of their parts, which typically excludes methods based on rigid iconic template matching.

Traditional shape-based recognition approaches fall into three broad categories: contour-based, morphological, and structural methods. All of these (with the exception of grey-scale morphology) require pre-segmentation of the image. Contour-based methods operate on the outline curve of the objects, which are difficult to obtain under real conditions and many objects do not have well-defined boundaries at all. Contours are typically obtained by edge- or region-based segmentation. Often, closed object contours are required for successful matching, which are even more difficult to extract in practice. Morphological methods are useful for simple shape-based filtering but are intolerant to rotation and scale changes. Binary morphology, in addition, requires prior segmentation (thresholding) as well. Structural recognition methods are based on the availability of primitive structural elements and the assumption that these elements can be extracted from the image data with sufficient reliability. Typical primitives are straight line segments, corners, blobs, arcs, and other parametric strokes. Regardless of what the primitives are, the performance of the recognition process depends critically on how reliably they can be extracted, which is difficult even under ideal viewing conditions. When images are noisy and cluttered, the extraction of suitable primitives may not be possible from local information alone. A typical example for this kind of imagery is shown in Figure 4.1. Also, the chosen class of primitives may significantly constrain the kind of objects that can be described.

Our goal is to implement an object indexing and recognition mechanism that makes effective use of *all* available sıructural information while, at the same time, avoids the early decisions necessary for extracting structural tokens. Instead of performing an isolated, context-independent detection and subsequent combination of structural primitives, larger spatial patterns are extracted and classified as a whole. The key mechanism we employ for this purpose is the hidden Markov model (HMM) which has been used successfully in other

Figure 4.1: Typical forward looking infrared image that contains rich structure but structural components are difficult to extract from.

pattern recognition applications, particularly for speech and character recognition. The main attraction of HMMs is that they are able to model possible pattern distortions, they perform efficient evidence accumulation without requiring previous segmentation, and they provide elegant learning methods. The main problem associated with traditional HMMs is that they are essentially one-dimensional models and their extension for 2-D applications is not straightforward. Our solution is to represent the object's appearance by a set of characteristic one-dimensional observation sequences that extend over the entire object (including its interior regions) and can be extracted from the image with sufficient reliability.

4.2 Hidden Markov Models for Signal-To-Symbol Conversion

The internal mechanism of the HMM is a probabilistic finite-state machine that is usually stationary, i.e. with fixed (time-invariant) state transition probabilities. The key difference from a regular Markov chain is that the state trajectory $s_0, s_1, \ldots, s_t, \ldots$ of a HMM is not directly observable. At each time step t, the HMM is thought of emitting an observable entity O_t, which can have either a discrete or a continuous value (for *discrete* or *continuous* HMMs, respectively). The observation O_t again depends upon the state s_t in a probabilistic way, which makes the model double-stochastic. Formally, a discrete HMM is defined as a tuple $\lambda = (A, B, \pi)$, where $A_{i,j}$ is the probability that state j follows state i, $B_{i,k}$ is the probability of observing symbol k in state i, and π_i is the probability of i being the initial state.

HMMs are well suited for handling two common difficulties in pattern recognition: spatial (or temporal) pattern distortions are captured by the non-deterministic internal finite-state machine, and the uncertainty of the locally observed signals are modeled by the observation probability distributions. During classification, the HMM uses all the available local information and interprets the subparts of a pattern *in the context* of a globally optimal solution. In case of the Viterbi classifier, we can obtain an implicit *segmentation* of the pattern into meaningful constituents from the corresponding optimal state trajectory, without the need for ever declaring explicitly what and where these constituents are. The considerable success of HMMs in many speech recognition applications is mainly due to these properties, in addition to their simplicity and efficiency.

While HMMs have a lot of appeal as a pattern recognition formalism, they are primarily suited for sequential, one-dimensional patterns and it is not obvious how HMMs can be applied for processing 2-D patterns. Markov Random Fields (MRF) and Hidden Markov Random Fields (HMRF) have been proposed as 2-D extensions of the original Markov chain model and the HMM, respectively. However, the MRF models lack efficient classification and learning algorithms and their impact in 2-D processing has been restricted to image restoration and pixel labeling. It has not been shown yet that MRFs are useful for structural pattern classification in the context of object recognition. Our approach is to apply HMMs in their original one-dimensional formulation by converting the image into a suitable sequence of observations.

The two key issues related to HMM algorithms for our purpose are (a) the classification of a given observation sequence, and (b) training the model from sample observations. In the *classification problem*, we are looking for the likelihood that a given observation sequence $\mathbf{O} = \langle O_0, O_1, \ldots O_{T-1} \rangle$ was generated by a particular HMM λ. Fortunately, there are simple and efficient algorithms for computing this likelihood, such as the *Forward-Backward* procedure and the classical *Viterbi* algorithm [106, 84]. Both algorithms are based on the dynamic programming principle and are linear in the length of the observation sequence for a given model. The Forward-Backward procedure computes the overall probability $P(\mathbf{O}|\lambda)$ for all possible state sequences $\mathbf{S} = \langle s_0, s_1, \ldots s_{T-1} \rangle$, while the Viterbi algorithm computes the probability of the "best" of all possible state sequence $\max_l P(\mathbf{O}|\lambda, \mathbf{S}_l)$. The *training problem* consists of finding a HMM λ' that maximizes the likelihood of observing a given set of observation sequences $\{\mathbf{O}^{(q)}\}$. While there is no analytical solution to this problem, an iterative learning procedure with proven convergence properties exists in the form of the *Baum-Welch* reestimation algorithm [84].

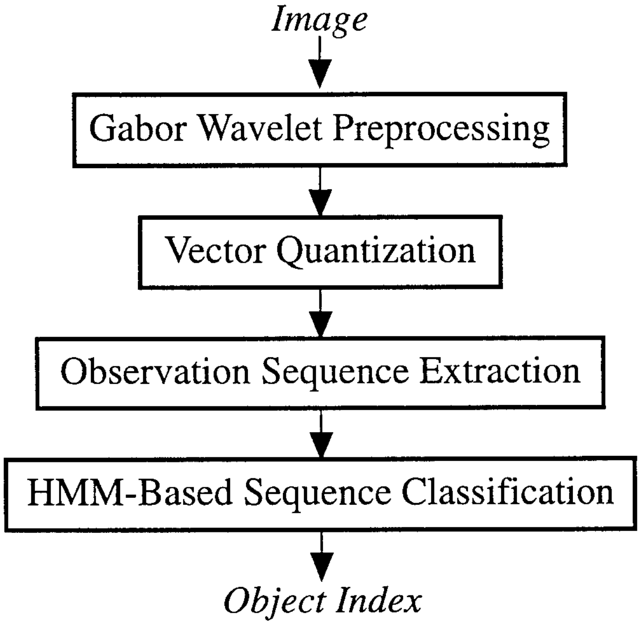


Image
Gabor Wavelet Preprocessing
Vector Quantization
Observation Sequence Extraction
HMM-Based Sequence Classification
Object Index

Figure 4.2: Principal components of the HMM-based signal-to-symbol conversion approach for object indexing.

4.3 Indexing Approach

Figure 4.2 shows the principal components of our approach. The input of the HMM-based structural analysis process is obtained from a set of oriented Gabor filters, which form a multi-scale decomposition of the image signal. Subsequently, the Gabor feature vectors are encoded to a discrete set of symbols by vector quantization (VQ). This is followed by the extraction of 1-D sequences from the 2-D image data, for which we describe several alternatives. Our approach is to extract observation sequences using both image and spatial constraints. HMMs are used to describe the shapes of individual subparts of objects, thus allowing recognition under partial occlusion. The identification of subparts is used for indexing into a given model base, which is the basis for final recognition. Both the VQ codebook and the HMM model parameters are learned directly from image examples in a supervised fashion.

4.3.1 Gabor Wavelet Features

Preprocessing of the image data consists of applying a set of hierarchical Gabor wavelet filters with N_ω log-spaced center frequencies ω_k and N_ϕ regularly spaced orientations ϕ_l. For each pixel location $\mathbf{x}_i$, we compute the values

$$\begin{aligned} J^+_{k,l}[i] &= (I * G^+_{\omega_k,\phi_l})[\mathbf{x}_i] \\ J^-_{k,l}[i] &= (I * G^-_{\omega_k,\phi_l})[\mathbf{x}_i] \end{aligned}$$

where I denotes the original image, $(G^+_{\omega_k,\phi_l}, G^-_{\omega_k,\phi_l})$ is a Gabor quadrature filter pair with center frequency ω_k and modulation orientation ϕ_l, and $*$ is the convolution operator. $G^+_{\omega_k,\phi_l}$ and $G^-_{\omega_k,\phi_l}$ are the cosine and the sine Gabor filter kernels, respectively. The spacing in frequency is Δ_ω, such that $\omega_k = \omega_0 \cdot \Delta^k_\omega$ for $1 \leq k \leq N_\omega$, and the spacing in orientation is $\Delta_\phi = \frac{\pi}{N_\phi}$, such that $\phi_l = \phi_0 + l \cdot \Delta_\phi$ for $1 \leq l \leq N_\phi$, for given ω_0, N_ω, ϕ_0, and N_ϕ. For typical values of $N_\phi = 4$, $\Delta_\omega = 2$, and $N_\omega = 4$, we obtain a 32-element *Gabor probe* $\mathbf{G}[i]$ for each image location $\mathbf{x}_i$. Figure 4.3 shows the cosine and sine responses of four filters with different center frequencies but identical orientation applied to the image in Figure 4.1.

Each individual Gabor probe at an image location $\mathbf{x}_i$ is a multi-scale description of the structural image properties around the point $\mathbf{x}_i$. The diameter of the region covered by each Gabor filter is inversely proportional to the filter's center frequency ω_k. Thus in a sense, the "larger" filters provide the structural context for the smaller ones. Notice that all filters are applied to each image position, which means that at least for the low-frequency Gabor filters the resulting output function is highly over-sampled. While the amount of data is considerably larger than with a traditional pyramid representation, the advantage of this scheme is that there is no need for spatial interpolation.

4.3.2 Encoding of Image Data

To produce the observation sequences required by a discrete HMM, we apply vector quantization to the continuous-valued Gabor probes. Each Gabor feature vector $\mathbf{y} = \mathbf{G}[i]$ is mapped to a discrete set of symbols: $\mathbf{y} \rightarrow q(\mathbf{y}) \in [0, L-1]$, for a L-level *codebook*. The issues in VQ are the choice of a suitable distance (distortion) measure, the codebook design, and the encoding efficiency. For the choice of a distance measure we have two basic alternatives: (a) group the Gabor probes by using the Euclidean (or similar) distance measure or (b) use a "smart" inter-vector distance measure that takes care of Gabor probe similarities over varying rotation and scale.

In the first approach (which we use), the burden of providing scale and rotation invariance lies on the subsequent HMM processing. In the second case, we would need a much more sophisticated distance measure. Given two Gabor probes a and b, the task would be to find the maximum similarity between a and b under all possible rotations and scale changes. For the chosen vector structure, a scale change would roughly be equivalent to a blockwise shift of elements up or down the vector, while a change in orientation would correspond to a simultaneous cyclic shift within each of the scale blocks. Clustering is used for the codebook design. Using the Euclidean distance in the multi-dimensional Gabor feature space, $d(\mathbf{y}, \hat{\mathbf{y}}) = ||\mathbf{y} - \hat{\mathbf{y}}||^2$ allows the use of k-means and similar clustering algorithms for building the codebook. Currently we are use a linear codebook consisting of $L = 128$ cluster entries, which will be replaced by a larger tree-structured codebook for faster quantization in the future. Figure 4.4 illustrates the result of applying VQ to the Gabor decomposition partly shown in Figure 4.3.

4.3.3 Sequentialization of Image Probes

The main problem is how to use the HMM formalism, which is geared towards one-dimensional sequence analysis, in the 2-D domain. One solution is the use of the objects' contours, which is particularly practical when the contours are closed [51], or the "pseudo-2D" version of HMMs that was used in [3] for printed character recognition. Both solutions for sequentializing image observations are not adequate for our problem.

The sequences we want to generate should (a) extend over the interior of the object, (b) they should be based on local image features (coded Gabor probes) without previous segmentation, and (c) they should not require the object to be entirely visible for successful indexing. Our model-base consists of a set of *sequence prototypes*, each represented by its corresponding HMM. The characteristic sequences for each object (or object view) are learned from real images. For model indexing, observation sequences are extracted from the given image and matched to the set of sequence prototypes. However, a key problem is *how* to find sequences of image locations during indexing that have a high likelihood of matching those stored in the HMM model base. Assuming that we are given (from the preceding detection stage, e.g.) a region of interest supposed to contain a known object, then some of the alternatives for producing observation sequences are:

1. *Random walk* — Start at an arbitrary location within the search region and perform a constrained random walk covering the search region. Although there is a chance that this process will eventually produce subsequences that coincide with existing model sequences, this technique is not efficient.

2. *Spatial constraints* — Using spatial constraints is one way of increasing the likelihood that sequences recovered from the region of interest have a match in the model base.

A very simple spatial constraint would be to move only horizontally, vertically, and diagonally over single pixel increments, regardless of the image contents. This creates problems with rotated patterns.

3. *Image constraints* — An alternative to using the underlying image raster, one could use the information in the image itself for constraining the sequentialization process. For example, one could extract local energy peaks as "control points" of the sequences and use a fixed but non-deterministic decision rule to move from one control point to the next. One could then use either Gabor information at the control points alone or include additional data between those points as we do here.

4. *Model constraints* — To further increase the likelihood of generating sequences that match the model one can make the sequencing itself model-dependent. In this scheme, the decision which successor probe to select depends upon the state of one or more HMMs in the model base. *This means that the sequence selection and matching steps become intimately coupled.* One way of formalizing this combined process is a Markov Decision process. In this case we can still use dynamic programming as in the original Viterbi decoding algorithm, but the sequence selection adds a spatial dimension to the trellis search space (see [12] for details).

Our current implementation is based on a combination of spatial constraints and image constraints. Image constraints are used to locate potential terminal points of Gabor probe sequences. In particular, we use local maxima of the Gabor energy $E_G[i] = ||\mathbf{G}[i]||^2$ to isolate such candidate points. Then, pairs of potential terminal points are connected by a straight line and the Gabor probes along this line are collected from the encoded (vector-quantized) data into discrete observation sequences, which we call Gabor *streaks*. The straight line serves as a simple spatial constraint for extracting the observation sequences that is independent of the image contents. The process for selecting pairs of terminal points is based on a model-independent heuristic decision rule that takes into account (a) the local Gabor energy value, (b) the distance between the points, (c) the number of streaks ending in a single point, and (d) the resulting local density of Gabor streaks. Of course, the same rule is applied during learning and recognition. Redundancy in the object representation is supposed to compensate for the uncertainties involved in this bottom-up process. Figure 4.5 shows an example for terminal point and subsequent Gabor streak extraction.

4.3.4 Sequence Classification and Indexing

In our current implementation, the selection of observation sequences is independent of the sequence classification. The extracted Gabor streaks are fed into all HMMs and the model that maximizes the probability of observing the given sequence is selected as the best match (Figure 4.6). If the maximum probability is below a certain threshold, the observation sequence is classified as "unknown". The use of dynamic programming schemes, such as the Viterbi algorithm, makes this process computationally efficient. Moreover, the matching could proceed in parallel for all HMMs.

4.4 HMM Model Base

For the purpose of indexing, an object aspect is represented by a set of Gabor streaks, each describing a certain part of the object's appearance. Indexing is accomplished by associating one or more Gabor streaks within a given region of interest with a particular object aspect, thus allowing indexing under partial object occlusion. The particular type of HMM used for representing Gabor streaks has a typical forward structure with five states, as shown in Figure 4.7. This type of model enforces an ordered left-to-right state sequence but allows individual states to be extended over several observation frames or to be skipped.

Gabor streak HMMs are acquired in a supervised learning process from real image data. For a given pair of terminal points, the model for the corresponding Gabor streak is computed by (a) randomly disturbing the coordinates of the terminal points, (b) extracting the corresponding disturbed streaks, and (c) feeding them into the Baum-Welch reestimation algorithm. This makes the model tolerant against positioning errors of the terminal points during indexing. For example, 30 learning trials on the streak connecting the terminal points 1 and 7 in Figure 4.5(b) using the codebook shown in Figure 4.4(c) resulted in the state transition probability matrix

$$A^{(1,7)} = \begin{bmatrix} 5 \cdot 10^{-11} & 0.999 & 0.5 \cdot 10^{-6} & 1 \cdot 10^{-32} & 0.0 \\ 3 \cdot 10^{-27} & 0.647 & 0.353 & 9 \cdot 10^{-7} & 0.0 \\ 6 \cdot 10^{-34} & 4 \cdot 10^{-18} & 0.055 & 0.945 & 0.0 \\ 3 \cdot 10^{-60} & 1 \cdot 10^{-36} & 7 \cdot 10^{-29} & 0.950 & 0.049 \\ 0.0 & 1 \cdot 10^{-68} & 2 \cdot 10^{-47} & 8 \cdot 10^{-24} & 1.0 \end{bmatrix}$$

which shows the left-to-right structure of the model and the emergence of distinct states.

4.5 Future Work

Our current work is focussed on the implementation of the model-dependent sequentialization process and a multi-aspect/multi-object HMM model base for target indexing and recognition. Near-term goals include the incorporation of multiple VQ codebooks and non-stationary HMMs to increase the robustness of the approach.

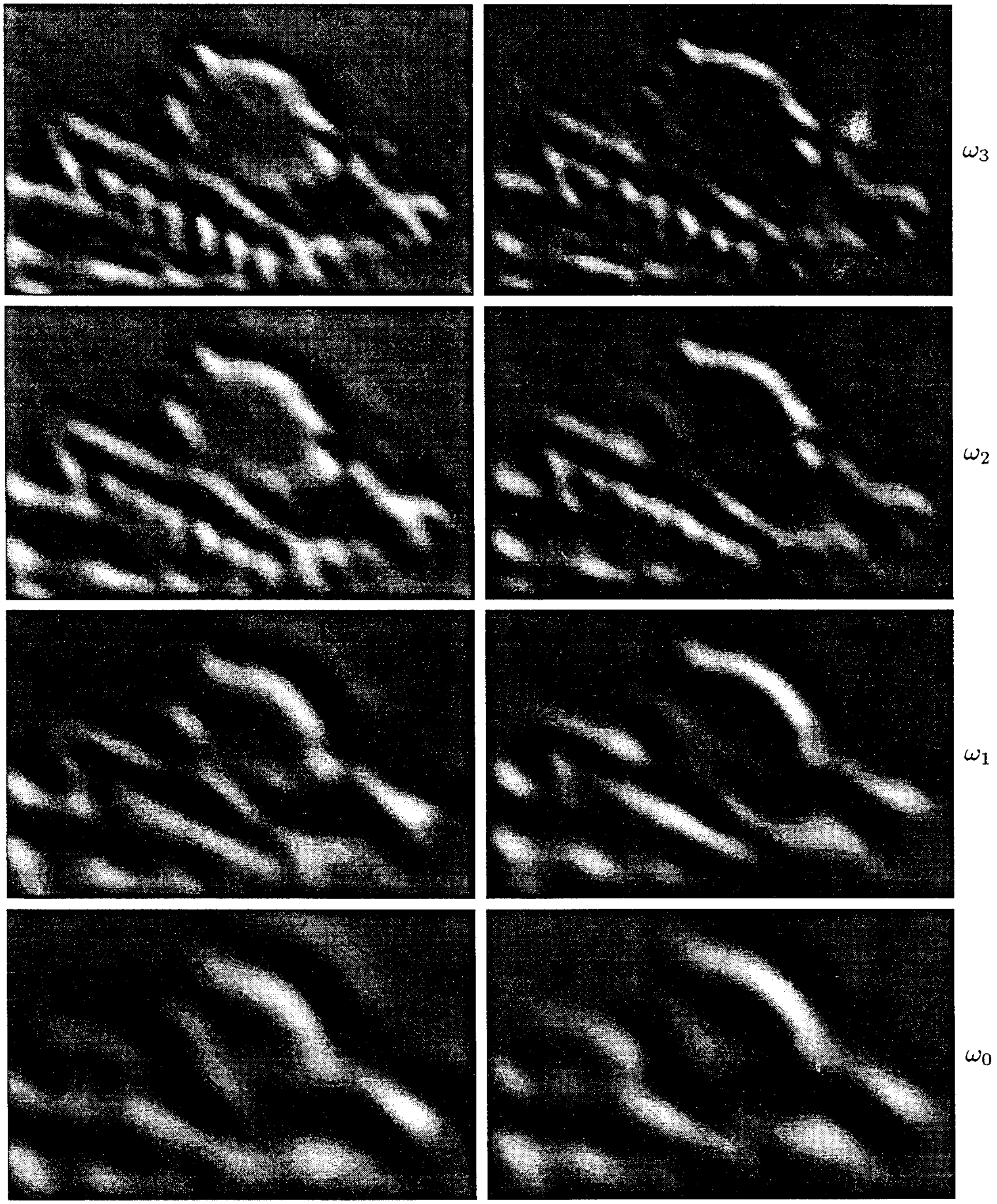


Figure 4.3: Cosine (left) and sine (right) components of the Gabor filter response for one out of four orientations ϕ_l and four different center frequencies $\omega_0 = 0.05\pi$, $\omega_1 = 0.1\pi$, $\omega_2 = 0.2\pi$, and $\omega_3 = 0.4\pi$.

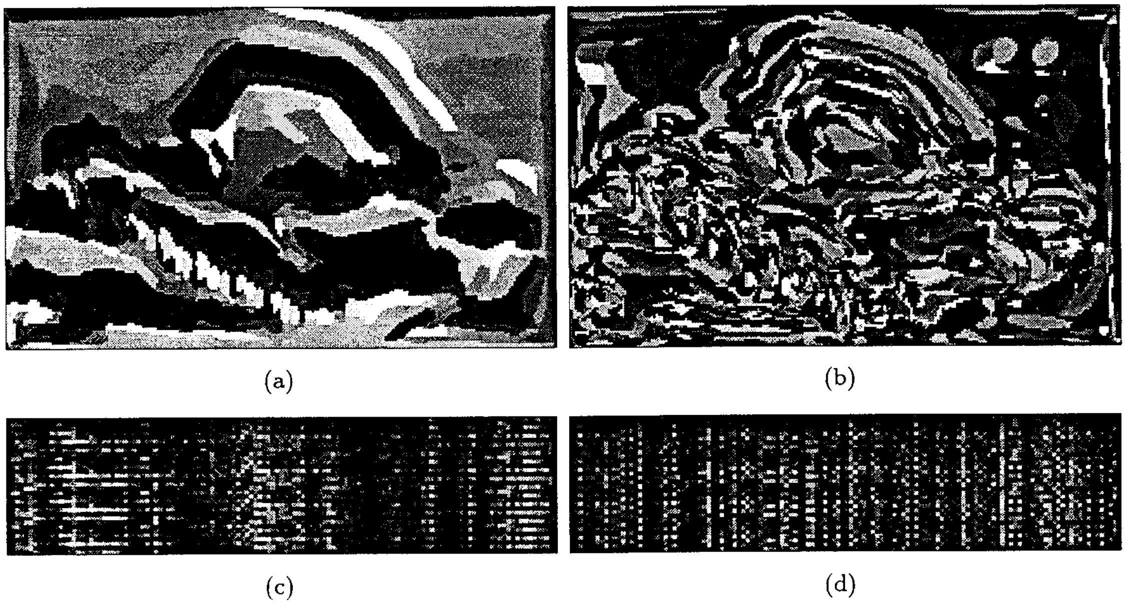
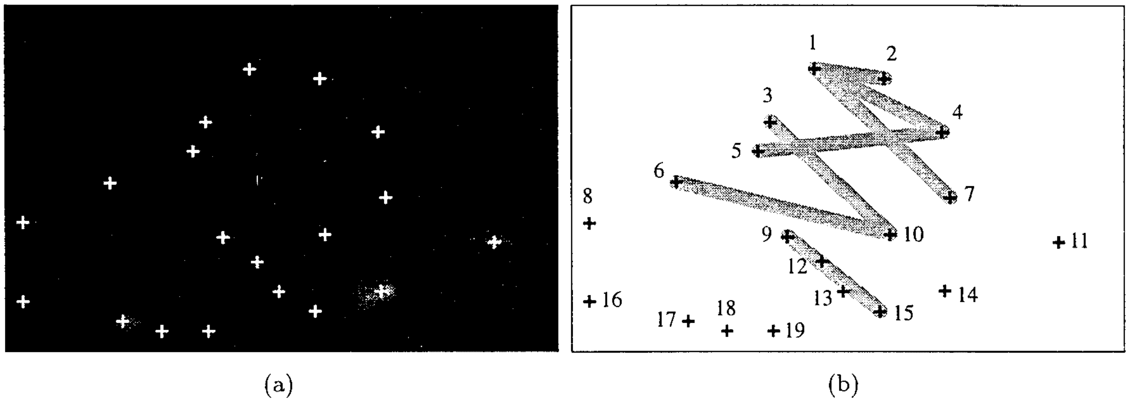


Figure 4.4: Result of vector quantization applied to the Gabor decomposition shown in Figure 4.3, using two different codebooks with 128 entries each (a–b). The corresponding 128 32-dimensional codebook vectors (c–d), where each vector is shown as a vertical column.



(a) (b)

Figure 4.5: Possible terminal points obtained at local maxima of the Gabor energy function (a). Gabor probe sequences (streaks) are formed by collecting encoded Gabor probes along straight lines between terminal points (b).

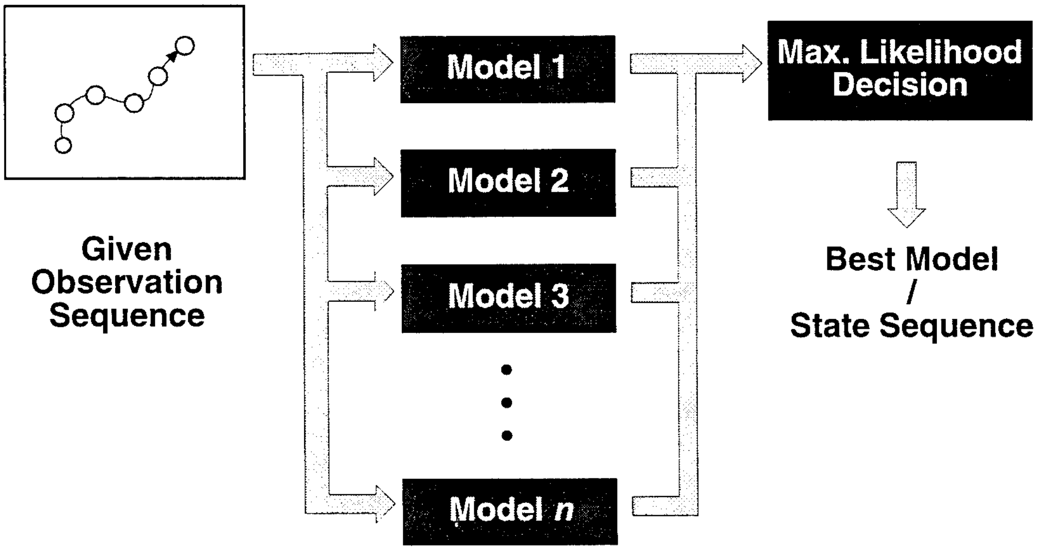


Figure 4.6: Using multiple HMMs for classifying an observation sequence, for the case that the sequentialization and classification are decoupled.

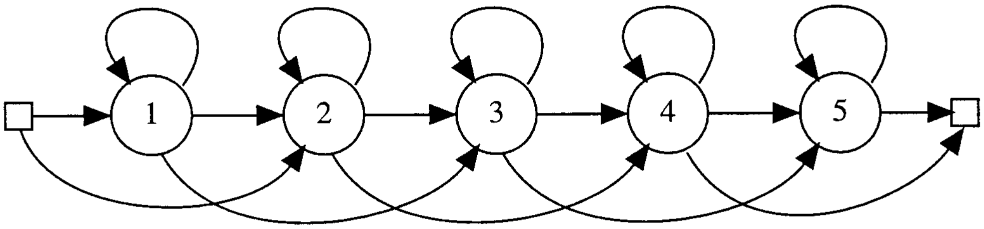


Figure 4.7: State transition diagram for the forward-type HMM used to represent Gabor probe sequences. State transitions not shown in the diagram are assigned zero probabilities.

Chapter 5

Closed-Loop Object Recognition Using Reinforcement Learning

5.1 Introduction

Image segmentation, feature extraction and model matching are the key building blocks of a computer vision system for model-based object recognition [27, 74]. The tasks performed by these building blocks are characterized as the low (segmentation), intermediate (feature extraction) and high (model matching) levels of computer vision. The goal of image segmentation is to extract meaningful objects from an image and it is essentially a pixel-based processing. Model matching uses a representation such as shape features obtained at the intermediate level for recognition. It requires explicit shape models of the object to be recognized. There is an abstraction of image information as we move from low to high levels and the processing becomes more knowledge based or goal directed.

Current computer vision algorithms for object recognition do not achieve good performance for practical applications since they do not adapt to the changing environment [14]. For object recognition systems to perform effectively under changing environmental conditions, it is essential to combine the interaction between the low and high level components of a vision system. There are several problems with current model-based object recognition systems that are mostly open-loop or *filter* type systems.

1. The fixed set of parameters used in various vision algorithms used for object recognition lead to undraceful degradation in performance.

2. The image segmentation, feature extraction and selection are generally carried out

as preprocessing steps to object recognition algorithms for model matching. These steps totally ignore the effects of the earlier results (image segmentation and feature extraction) on the future performance of the recognition algorithm.

3. Generally the criteria used for segmentation and feature extraction require elaborate human designs. When the conditions for which they are designed are changed slightly, these algorithms may fail. Furthermore, the criteria themselves can be a subject of debate [16].

4. Object recognition is a process of making sequences of decisions, i.e., applying various image analysis algorithms. Often the usefulness of a decision or the results of an individual algorithm can *only* be determined by the final outcome (e.g. matching confidence) of the recognition process. This is also known as "vision-complete" problem [26], i.e., one cannot really assign labels to the image without the knowledge of which parts of the image correspond to what objects.

Object recognition systems whose decision criteria for image segmentation and feature extraction, etc. are developed autonomously from a reinforcement signal of the final recognition might transcend all the above problems.

In this chapter, we present an approach that takes the output of the recognition algorithm and uses it as a feedback in a reinforcement learning framework to influence the performance of the algorithm used for image segmentation. The recognition performance is improved over time with this method. The novelty of the approach is that it includes the matching or recognition component as part of the evaluation function for image segmentation in a systematic way. The additional strength of the approach is that the system develops its independent decision criteria (segmentation parameters) to best serve the underlying recognition task. It should be emphasized that our interest is not in a simple mixture of learning and computer vision, but rather in the principled integration of the two fields at the algorithmic level.

Our work is most closely related to the work by Bhanu and Lee [16], where they describe a system that uses genetic algorithms for learning segmentation parameters[1]. However, the recognition algorithm is not part of the evaluation function for segmentation in their system. The genetic algorithms simply search for a set of parameters that optimize a prespecified evaluation function which may not best serve the overall goal of robust object recognition. Furthermore, they assume that the location of the object in the image is known. In our work, we do not make such an assumption. We use explicit geometric model of an object, represented by its polygonal approximation, to recognize it in the image.

[1] Specifically, the PHOENIX's [60] parameters.

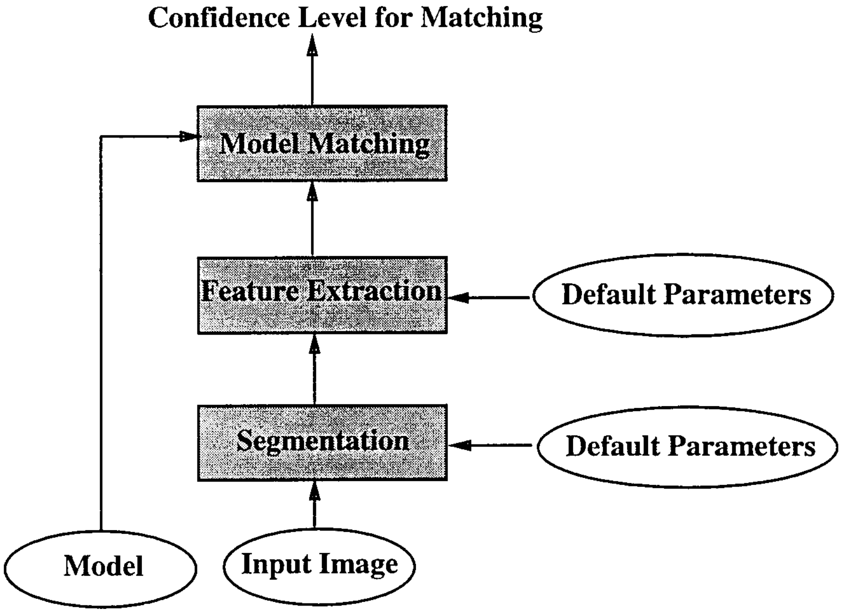


Figure 5.1: Conventional multi-level system for object recognition.

Section 5.2 describes a general framework for reinforcement learning-based adaptive image segmentation. Section 5.3 describes the reinforcement learning paradigm and the particular reinforcement learning algorithm employed in our system. Section 5.4 presents the experimental results evaluating the system and section 5.5 concludes the chapter. Appendices A and B describe briefly the segmentation and the matching algorithms that have been used to perform experiments reported in section er.

5.2 Reinforcement Learning System for Segmentation Parameter Estimation

5.2.1 The Problem

Consider the problem of recognizing an object in an input image, assuming that the model of the object is given and that the precise location of the object in the image is unknown.

The conventional method, shown in Figure 5.1, for the recognition problem is to first segment the input image, then extract and select appropriate features from the segmented image, and finally perform model matching using these features. If we assume that the matching algorithm produces a real valued output indicating the degree of success upon its completion, then it is natural to use this real valued output as feedback to influence the performance of segmentation and feature extraction so as to bring about system's earlier decisions favorable for more accurate model matching. The rest of the chapter describes a reinforcement learning-based vision system to achieve just that.

5.2.2 Learning to Segment images

Our current investigation into reinforcement learning-based vision systems is focused on the problem of learning to segment images. An important characteristic of our approach is that the segmentation process takes into account the biases of the recognition algorithm to develop its own decision strategies. As a result, more efficient recognition performance can be expected.

Image Segmentation

We begin with image segmentation [50] because it is an extremely important and difficult low-level task. All subsequent interpretation tasks including object detection, feature extraction, object recognition and classification rely heavily on the quality of the segmentation process. The difficulty arises for image segmentation when only local image properties are used to define the region-of-interest for each individual objects. It is known [36] that correct localization may not always be possible. Thus, image segmentation cannot be done by grouping parts with similar image properties in a purely bottom-up fashion. Difficulties also arise when segmentation performance needs to be adapted to the changes in image quality, which is affected by variations in environmental conditions, imaging devices, time of day, etc. The following are the key characteristics [16] of the image segmentation problem:

- When presented with a new image, selecting the appropriate set of algorithm parameters is the key to effectively segmenting the image.
- The parameters within most segmentation algorithms typically interact in a complex, non-linear fashion, which makes it difficult or impossible to model the parameters' behavior analytically.
- The variations between images cause changes in the segmentation results, the objective function that represents segmentation quality varies from image to image. Also, there may not be a consensus on segmentation quality measures.

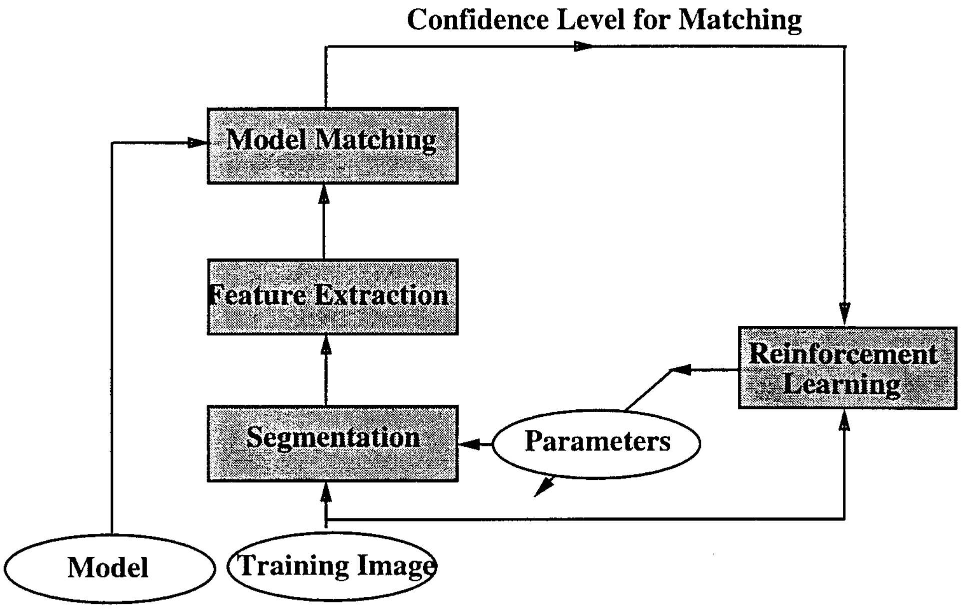


Figure 5.2: Reinforcement learning-based system for object recognition.

Our Approach

Figure 5.2 depicts the conceptual diagram of our reinforcement learning-based object recognition system that addresses the parameter selection problem encountered in image segmentation task by using the recognition algorithm itself as part of the evaluation function for image segmentation. Note that the reinforcement learning component employs a particular reinforcement learning algorithm which will be described in the next section. Figure 5.3 shows the main steps of the algorithm we use, where the algorithm terminates when either the number of iterations reaches a prespecified value (N) or the recognition confidence level (r) has exceeded a given threshold, called R_{th}. In the event that the number of iterations has exceeded N, we will say that the object is not present in the image. Also for simplicity we assume that only one instance of the model is present in the image. Multiple instances of the model can be recognized by slight modification of the algorithm.

```
• LOOP:
    1. For each image $i$ in the training set do
        (a) Segment image $i$ using current segmentation parameters
        (b) Perform noise clean up
        (c) Get segmented regions (also called blobs or connected components)
        (d) Perform feature extraction for each blob to obtain token sets
        (e) Compute the matching of each token set against stored model and return the highest confidence level, $r$
        (f) If $r \geq R_{th}$ then exit
        (g) Obtain new parameters for the segmentation algorithm using $r$ as reinforcement for the reinforcement learning algorithm
• UNTIL number of iterations is equal to N
```

Figure 5.3: Main Steps of the Reinforcement Learning-Based Object Recognition Algorithm.

5.3 Reinforcement Learning

In this section we begin with a brief overview of the reinforcement learning technique. We then describe reinforcement learning algorithms applicable to our task and the modifications of these algorithms to effectively solve the problem identified in section 5.2.1.

Reinforcement learning is a framework for learning to make sequences of decisions in an environment. In this framework, a learning system is given at each time step inputs describing its environment. The system then makes a decision based on these inputs, thereby causing the environment to deliver to the system a reinforcement. The value of this reinforcement depends on the environmental state, the system's decision, and possibly random disturbances. In general, reinforcement measuring the consequences of a decision can emerge at a multitude of times after the decision is made. A distinction can be made between associative and non-associative reinforcement learning. In the non-associative paradigm, reinforcement is the only information the system receives from its environment. Whereas, in the associative paradigm, the system receives input information that indicates the state of its environment as well as reinforcement. In such learning systems, a "state" is a unique representation of all previous inputs to a system. In computer vision, this state information corresponds to current input image and our object recognition applications require us to take into account the changes appearing in the input images. The objective of the system is to select sequences of decisions to maximize the sum of future reinforcement (possibly discounted) over time. It is interesting to note that for a given state an associative reinforcement learning problem becomes a non-associative learning problem.

As noted above, a complication to reinforcement learning is the timing of reinforcement. In simple tasks, the system receives, after each decision, reinforcement indicating the goodness of that decision. Immediate reinforcement occurs commonly in function optimization tasks. A well-understood method in immediate reinforcement learning is the REINFORCE algorithms of Williams [108], a class of connectionist reinforcement learning algorithms, that perform stochastic hill-climbing.

In more complex tasks, however, reinforcement is often temporally delayed, occurring only after the execution of a sequence of decisions. Delayed reinforcement learning is important because in many problem domains, immediate reinforcement regarding the value of a decision may not always be available. For example, in object recognition, the goodness of segmentation is not known until the recognition decision has been made. Delayed reinforcement learning is attractive and can play important role in machine vision.

The most effective approach to date to delayed reinforcement learning is the temporal difference learning method of Sutton [105], a class of useful computational procedures for solving the temporal credit assignment problem. The key idea behind temporal difference learning is that the value of a state should be regressed towards the weighted average of the values of its successors, where the weightings reflect the conditional probabilities of the successors. A well-studied form of temporal difference learning is Watkins' Q-learning [107], a Monte-Carlo method that approximates dynamic programming. For further details, see [107, 77]. The REINFORCE algorithms are the main focus in this chapter since reinforcement (matching confidence level) regarding segmentation performance is immediate.

5.3.1 REINFORCE Algorithms

The particular class of reinforcement learning algorithms employed in our object recognition system is the REINFORCE algorithms. It is a class of connectionist reinforcement learning algorithms developed by Williams [108], where units in such a network are *Bernoulli quasi-*

linear units, in that the output of such a unit is either 0 or 1, determined stochastically using the Bernoulli distribution with parameter $p = f(s)$, where f is the logistic function,

$$f(s) = 1/(1 + \exp(-s))$$

and

$$s = \sum_i w_i x_i$$

is the usual weighted summation of input values to that unit. For such a unit, p represents its probability of choosing 1 as its output value. Figure 5.4 depicts such a unit.

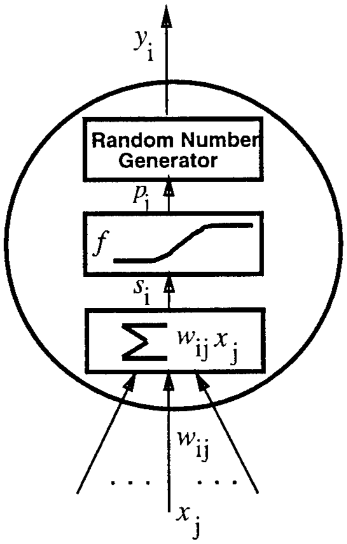


Figure 5.4: Bernoulli Quasilinear Unit

In the general reinforcement learning paradigm, the network generates an output pattern and the environment responds by providing the reinforcement r as its evaluation of that output pattern, which is then used to drive the weight changes according to the particular reinforcement learning algorithm being used by the network. For the Bernoulli quasilinear units used in this research, the REINFORCE algorithm we use prescribes weight increments equal to

$$\Delta w_{ij} = \alpha_{ij}(r - b_{ij})(y_i - p_i)x_j, \tag{5.1}$$

where α_{ij} is a positive learning rate (possibly different for each weight), b_{ij} serves as a *reinforcement baseline* (which can also be different for each weight), and $i = 1, \ldots, n,$

$j = 1, \ldots, m$. n and m are the number of the units in the network and the number of features for each input, respectively. In this chapter, we consider only algorithms having the form

$$\Delta w_{ij} = \alpha(r - b)(y_i - p_i)x_j \tag{5.2}$$

where $\alpha_{ij} = \alpha$ and $b_{ij} = b$ for all i and j. x_j is the input to the Bernoulli unit and y_i is the output of the ith Bernoulli unit. p_i is an internal parameter to a Bernoulli random number generator and it is computed according to (5.4). It can be shown [108] that, regardless of how b is computed, whenever it does not depend on the immediately received reinforcement value r, such an algorithm satisfies

$$E\{\Delta \mathbf{W}|\mathbf{W}\} = \alpha \nabla_{\mathbf{W}} E\{r|\mathbf{W}\} \tag{5.3}$$

where E denotes the expectation operator, $\mathbf{W}$ represents the weight matrix of the network, and $\Delta\mathbf{W}$ is the change of the weight matrix. A reinforcement learning algorithm satisfying (5.3) can be loosely described as having the property that it statistically climbs the gradient of expected reinforcement in weight space. For extensive discussions of these algorithms, see [108, 109]. Next two subsections describe the particular network and the REINFORCE algorithm used in the experiments reported in this chapter.

5.3.2 The Team Network

We use a very simple form of trial generating network in which all of the units are output units and there are no interconnections between them. This degenerate class of network corresponds to what is called a *team* of automata in the literature on stochastic learning automata [72]. We, therefore, call these networks as *teams of Bernoulli quasilinear units.* Figure 5.5 depicts the team network used here, which corresponds directly to the reinforcement learning component in Figure 5.2. Each segmentation parameter is represented by a set of Bernoulli quasilinear units and the output of each unit is binary as we have described earlier.

For any Bernoulli quasilinear unit, the probability that it produces a 1 on any particular trial given the value of the weight matrix $\mathbf{W}$ (size n by m)is

$$Pr\,\{y_i = 1|\mathbf{W}\} = p_i = f(s_i) = \frac{1}{1 + e^{-s_i}} \tag{5.4}$$

where $s_i = \sum_j w_{ij}x_j$. Because all units pick their outputs independently, it follows that for such a team of Bernoulli quasilinear units the probability of any particular output vector $\mathbf{y}(t)$, corresponding to an instance of segmentation parameters, conditioned on the current

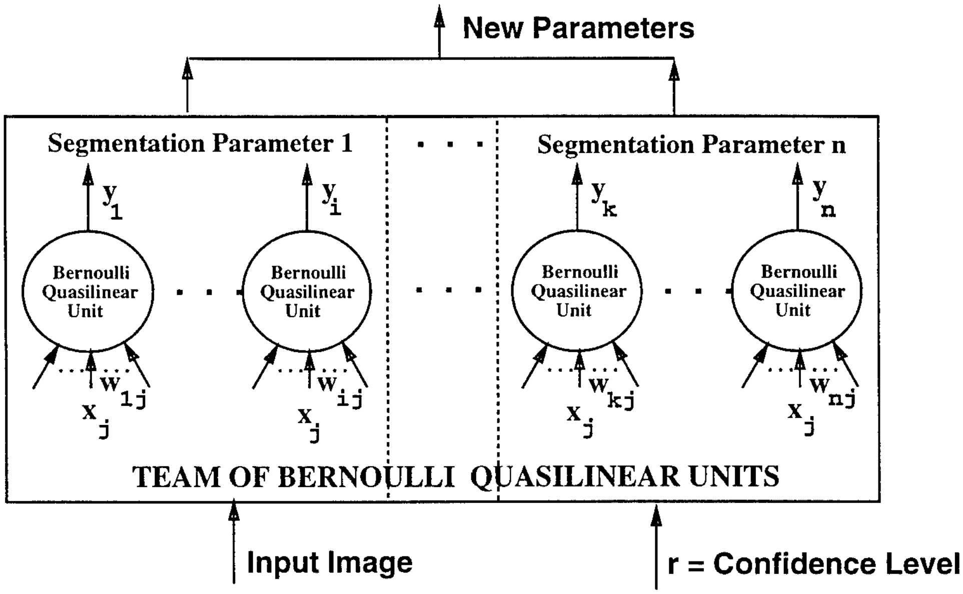

New Parameters
Segmentation Parameter 1
Segmentation Parameter n
y1
yi
yk
yn
Bernoulli Quasilinear Unit
Bernoulli Quasilinear Unit
Bernoulli Quasilinear Unit
Bernoulli Quasilinear Unit
w1j
wij
wkj
wnj
xj
xj
xj
xj
TEAM OF BERNOULLI QUASILINEAR UNITS
Input Image
r = Confidence Level

Figure 5.5: Team of Bernoulli units for learning segmentation parameters.

value of the weight matrix **W** is given by

$$Pr\,\{\mathrm{y}|\mathrm{W}\} = \prod_{i\in\{1,\cdots,n\}} p_i^{y_i}(1-p_i)^{1-y_i}. \tag{5.5}$$

The weights w_{ij} are adjusted according to the particular learning algorithm used. We note that when $s_i = 0$ and hence $p_i = 0.5$, thus the unit is equally likely to pick y_i either 0 or 1, while increasing s_i makes a 1 more likely. Adjusting the weights in a team of Bernoulli quasilinear units is thus tantamount to adjusting the probabilities (p_i's) for the individual units.

Note that, except bias terms, there are no input connections in the team networks experimented in [109]. In contrast, the team network described here does have input weights which play the role of long-term memory in associative learning tasks.

5.3.3 The Team Algorithm Used

The algorithm we used with the team architecture has the following general form: At the t^{th} time step, after generating output $\mathbf{y}(t)$ and receiving reinforcement $r(t)$, i.e., the confidence level indicating the matching result, increment each weight w_{ij} by

$$\Delta w_{ij}(t) = \alpha\rho(t)e_{ij}(t)x_j - \delta w_{ij}(t), \tag{5.6}$$

where α, the learning rate, and δ, the weight decay rate, are parameters of the algorithm. ρ is called the *reinforcement factor* and e_{ij} the *eligibility* of the weight w_{ij} [108]. Generally, the eligility of a weight indicates the extent to which the activity at the input of the weight was connected in the past with unit output activity. The reinforcement factor is computed according to

$$\rho(t) = r(t) - \bar{r}(t-1), \tag{5.7}$$

where $\bar{r}(t)$ is the exponentially weighted average, or *trace*, of prior reinforcement values

$$\bar{r}(t) = \gamma\bar{r}(t-1) + (1-\gamma)r(t) \qquad t \geq 1, \tag{5.8}$$

with $\bar{r}(0) = 0$. The trace parameter γ was set equal to 0.9 for all the experiments reported here. Finally, we considered the following form of eligibility

$$e_{ij}(t) = y_i(t) - \bar{y}_i(t-1), \tag{5.9}$$

where $\bar{y}_i(t)$ is an average of past values of y_i computed by the same exponential weighting scheme used for $\bar{r}$. That is,

$$\bar{y}_i(t) = \gamma\bar{y}_i(t-1) + (1-\gamma)y_i(t). \tag{5.10}$$

Superior performance with this form of eligibility was reported in the experiments performed in [109] for function optimization. For other forms of eligibility, see [109].

The use of weight decay is chosen as a simple heuristic method to force sustained exploration of the weight space since it was found that REINFORCE algorithms without weight decay always seemed to converge. It is argued in [109] that having weight decay (the second term $\delta w_{ij}(t)$ in Equation (5.6) is very closely related to having a nonzero mutation rate at a particular allele (feature value) in a genetic algorithm [44]. The size of the weight decay rate δ was chosen to be 0.01 in all our experiments. Note that there are other ways to force sustained exploration. One possibility is to maximize a linear combination of system's entropy and reinforcement. We omit here the detailed analysis of the method except commenting that such a strategy seeks not only a particular region of the space having high reinforcement values, but also a variety of such high value regions. For further details, see [109].

5.3.4 Implementation of the Algorithm

A slightly different training strategy from that described in Figure 5.3 was used in the experiments reported here. Instead of looping through every images in the training set, the training procedure samples images proportional to the level of matching confidence the current system achieves. That is, the lowerer the matching confidence the system gets on an image, the more likely the image will be sampled. In this way training is focused on those images having the lowest matching confidence, and thus faster performance improvement can be achieved. A similar technique is also adopted in [33]. Figure 5.6 shows the main steps of the proportional training algorithm, where $MAXCONFID$ is the maximum confidence level the system can achieve, i.e., when a perfect matching occurs.

5.4 Experimental Results

This section describes experimental results evaluating the performance of our system on two sets of color images, one of which is indoor and the other is outdoor.

The *PHOENIX* algorithm [60] was chosen as the image segmentation component in our system because it is a well-known method for the segmentation of color images with a number of adjustable parameters. It has been the subject of several Ph.D. theses [82, 102]. PHOENIX works by splitting regions using histogram for color features. Appendix 5.5 provides a brief overview of the algorithm. Note that any segmentation algorithm with adjustable parameters can be used in our approach.

The PHOENIX algorithm has a total of fourteen adjustable parameters. The four most critical ones that affect the overall results of the segmentation process are used in learning. These parameters are *Hsmooth*, *Maxmin*, *Splitmin*, and *Height*. *Hsmooth* is the width of the histogram smoothing window, where smoothing is performed with a uniformly weighted moving average. *Maxmin* defines the peak-to-valley height ratio threshold. Any interval (see Appendix 5.5) whose peak height to higher shoulder ratio is less than this threshold is merged with the neighbor on the side of the higher shoulder. *Splitmin* defines the minimum size for a region to be automatically considered for splitting. This is an absolute value, not a percentage of the image area. *Height* is the minimum acceptable peak height as a percentage of the second highest peak. The team algorithm searches for a combination of these parameters that will give rise to a segmentation from which the best recognition can be achieved.

The ranges for each of these parameters are the same as those used in [16]. Table 5.1 shows sample ranges for each of these parameters. The resulting search space is about one million sample points.

```
• LOOP:
    1. For each image $i$ in the training set do
        (a) Compute matching confidence for image $i$: $CONFID_i$
        (b) $n_i = MAXCONFID - CONFID_i$
        (c) If $\sum_i n_i$ is 0, then terminate.
        (d) $proportion_i = \frac{n_i}{\sum_i n_i}$
    2. Sample image $i$ according to $proportion_i$ and do
        (a) Segment image $i$ using current segmentation parameters
        (b) Perform noise clean up
        (c) Get segmented regions (also called blobs or connected components)
        (d) Perform feature extraction for each blob to obtain token sets
        (e) Compute the matching of each token set against stored model
            and return the highest confidence level, $r$
        (f) If $r \geq R_{th}$ then exit
        (g) Obtain new parameters for the segmentation algorithm using
            $r$ as reinforcement for the reinforcement learning algorithm
• UNTIL number of iterations is equal to N
```

Figure 5.6: Main Steps of the Proportional Training Algorithm.

Each of the PHOENIX parameters is represented using 5 bit Gray code which has the advantage over simple binary code in that only one bit changes between representations of two consecutive numbers. One reason for using the binary representation is its usefulness as a model of certain types of distributed adaptive decision-making. Another reason is that it

Table 5.1: **Sample ranges for selected PHOENIX parameters.**

Parameter	Sampling Formula	Test Range
Hsmooth: hsindex $\in [0 : 31]$	hsmooth=1 + 2 * hsindex	1 – 63

Maxmin:
mmindex $\in [0 : 31]$
	ep=ln(100) + 0.05 * mmindex
	maxmin = exp(ep) + 0.5
		100 – 471

Splitmin: smindex $\in [0 : 31]$	splitmin=9 + 2 * smindex	9 – 71
Height: htindex $\in [0 : 31]$	height=1 + 2 * htindex	1 – 63

offers a combinatorially advantageous way of approaching learning problems having a large search space. While the same task could be learned in the original parameter space, for many types of problems, including image segmentation, the binary representation can be expected to learn much faster. Since there are 4 parameters, we have a total of 20 Bernoulli quasilinear units and each parameter corresponds to the outputs of 5 units.

The feature extraction consists of finding polygon approximation tokens for each of the regions obtained after image segmentation. The polygon approximation is obtained using a split and merge technique [19] that has a fixed set of parameters.

Object recognition employs a cluster-structure matching algorithm [19] which is based on the clustering of translational and rotational transformation between the object and the model for recognizing 2-D and 3-D objects. A brief description of the algorithm is given in Appendix 5.5. The algorithm takes as input two sets of tokens, one of which represents the stored model and the other represents the input region to be recognized. It then performs topological matching between the two token sets and computes a real number that indicates the confidence level of the matching process. This confidence level is then used as a reinforcement signal to drive the team algorithm.

It is important to note that, in the current implementation of the system, the cluster-structure matching algorithm does not have the knowledge of actual target location in the image. It simply attempts to match the stored model against the polygonal approximation of each blob in the segmented image whose size is at least 80% of the size of the model, and at the same time does not exceed it by more than 20%. The confidence level returned is the highest value ever obtained during matching.

It is worth pointing out that, during learning, the weights are updated after each presentation of an input image. This is in direct analogy to the typical weight update procedure in connectionist networks where weights are updated according to the stochastic gradient or incremental procedure instead of the total gradient rule [61]. That is, updates take place after each presentation of a single exampler without averaging over the whole training set. Both empirical and theoretical studies show that the stochastic gradient rule converges significantly faster than the total gradient rule, especially when training set contains redundant information.

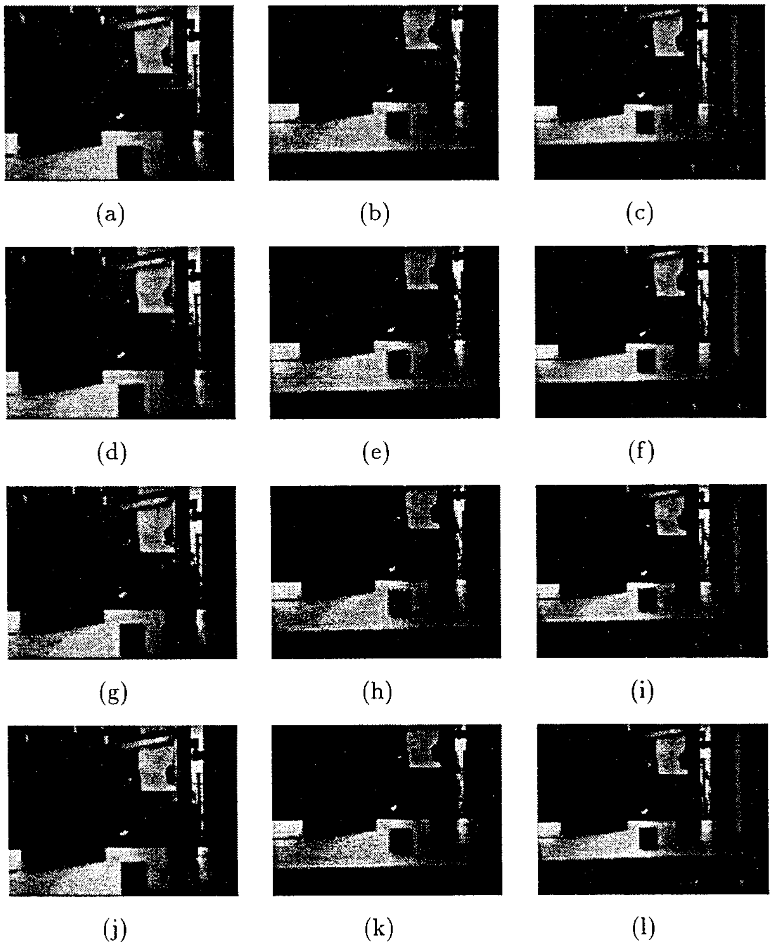
Finally, as a comparison, the segmentation results with the PHOENIX algorithm using default parameters are also obtained for feature extraction and recognition on the same tasks.

5.4.1 Results on Indoor Images

The first segmentation task whose experimental results we report here is a sequence of indoor color images (160 by 120 pixels) having simple geometric objects with varying lighting and motion conditions. These images are shown in Figure 5.7, where, from left to right, images are moving away from the camera, and within each column, lighting conditions deteriorate from top to bottom. The training data consist of the images in the first column (4 images), whereas the testing data come from the rest of the images (8 images). The objective of the task is to find a set of PHOENIX's parameters that give rise to a segmentation of the input image which, after appropriate feature extraction, will result in the recognition of the triangular object. The model of the triangular object is represented by a polygonal approximation of its shape. The threshold for matching confidence in this simple case was set to 100%. Note that, unlike previous work on image segmentation, the criteria measuring image segmentation quality here are completely determined by the matching algorithm itself.

Each unit in the team network has a total of 8 input weights, each of which takes an average gray value of input on a 60 by 40 neighborhood on the input image plane of 120 by 160 pixels. This input plane is the luminance image of the corresponding color image. Note that in this experiment the average is normalized to lie between -1 and 1. For weights that are adjacent in a unit, their receptive fields are at least 40 pixels apart in the input image. Thus, the input image is undersampled, which in turn greatly reduces the number of weights in the network. The motivation is that variations in lighting need not be adapted with high resolution.

Figure 5.8 shows the segmentation performance (both training and testing) of the PHOENIX algorithm with learned parameters on the images shown in Figure 5.7. The training results in Figure 5.8 are obtained after a mean value (over 10 runs) of 103 passes through the



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

(j) (k) (l)

Figure 5.7: Twelve color images having simple geometric objects.

training data. Figure 5.9 shows the average confidence received by the system over time during training. Figure 5.10 shows the trajectory of each of the four *Hsmooth*, *Maxmin*, *Splitmin*, and *Height* parameters during training in a typical run on a particular image (in this case it is the third image in the first column of Figure 5.7, i.e., 5.7(g)). Note that no attempt was made to determine if the set of parameters giving rise to the final recognition is unique.

When the segmentation parameters obtained after training were applied to the images in the testing set, recognition results for images 5.7(b), 5.7(c), 5.7(f), 5.7(i) and 5.7(k) are

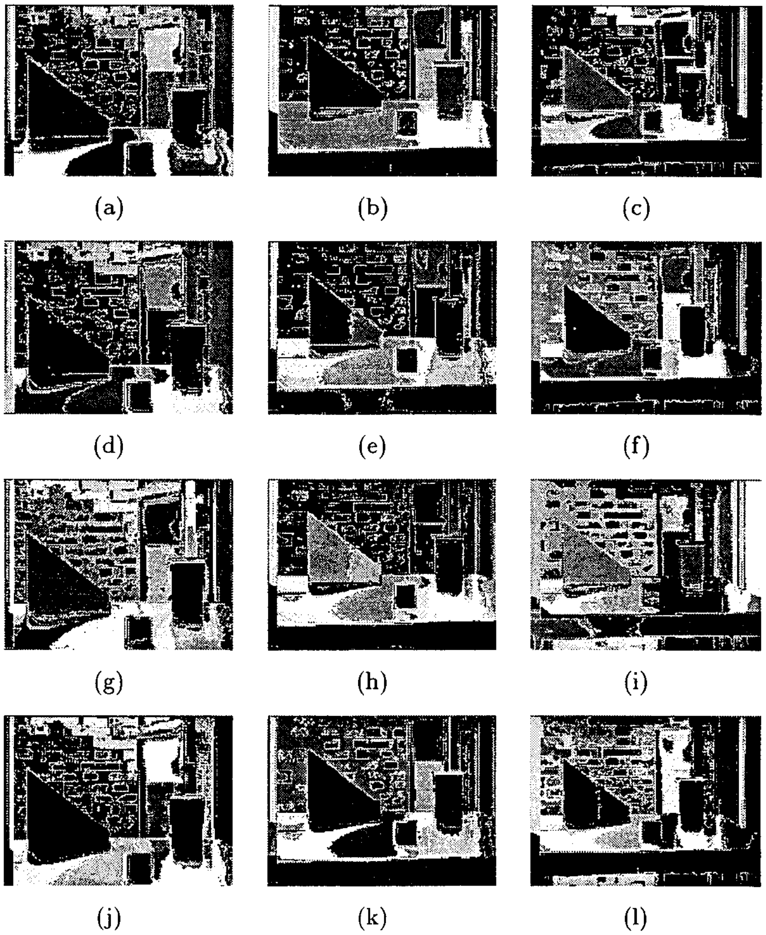


Figure 5.8: Segmentation performance of the PHOENIX algorithm with learned parameters.

acceptable. However, recognition failed for images 5.7(e), 5.7(h) and 5.7(l). If we allow learning to continue on these three images, experiments have been performed which show that successful recognition can be achieved for all testing images in much less time (less than 50%) compared to the time taken for training on the images shown in the first column of Figure 5.7.

In comparison, the PHOENIX algorithm with default parameter setting was also run

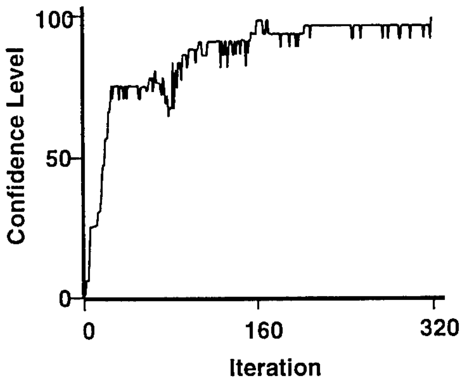


Figure 5.9: Average confidence received over time during training.

on the same images. Figure 5.11 shows the samples of the segmentation performance of the PHOENIX algorithm with default parameters on the images in the first row of Figure 5.7, i.e, images 5.7((a)), 5.7(b), and 5.7 (c). These default parameters were obtained in [60] after extensive tests. We omit the details of the experiment, but note that this default parameter setting resulted in a total matching failure.

5.4.2 Results on Outdoor Images

The second segmentation task involves a sequence of 10 outdoor color images obtained under varying environmental conditions, two of which are shown in Figure 5.12. These images are collected approximately every 15 minutes over $\sim$ 2 and 1/2 hour period [16]. The images exhibit varying shadow and reflection on the car as the position of the sun changed and clouds came in and out the field of view of the camera that had auto iris adjustment turned on. The overall goal is to recognize the car in the image. The original images are digitized at 480 by 480 pixels in size and are then subsampled to produce 120 by 120 pixel images. Five of these odd-numbered images are used as training data and five even-numbered images as testing data.

Similar to the team network for the indoor images, each unit here has a total of 9 input weights, each of which takes an average gray value of input on a 40 by 40 neighborhood on the input image plane of 120 by 120 pixels. These averages are normalized to lie between -1 and 1. Polygonal approximation of the car shown in Figure 5.13 is used as the model in the cluster-structure matching algorithm. It was extracted manually in an interactive session

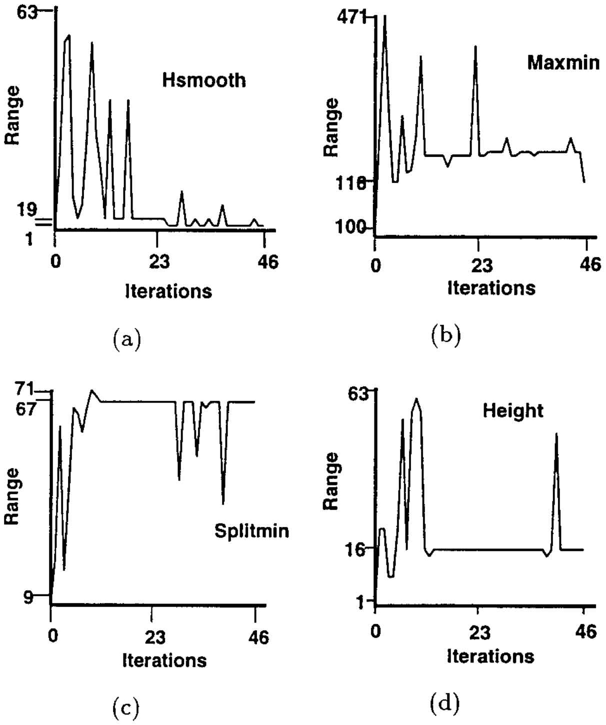


Figure 5.10: Trajectories for a particular run for each of the four parameters *Hsmooth*, *Maxmin*, *Splitmin*, and *Height* during training on a particular image (Figure 5.7(g)).

from the first frame in the sequence.

Figure 5.14 shows a sequence of segmentations for frame 1 with PHOENIX's parameters sampled at iterations 20, 30, 40, 50, 60, and 74 in a particular run during training, and corresponding parameter values at each of these intervals are shown in Table 5.2. Note that Figure 5.14(f) shows the final segmentation result when the highest confidence matching has been achieved. The threshold for acceptable matching confidence is set at 90% because of the low resolution nature of the real data.

Figure 5.15 shows the Phoenix segmentation performance on two testing images (frames 2 and 4) of with learned parameters obtained after training on frames 1, 3, 5, 7 and 9. For frame 2 the matching is acceptable. However, for frame 4 the result is not acceptable and learning is to be performed similar to the indoor examples for the adaptation of parameters.

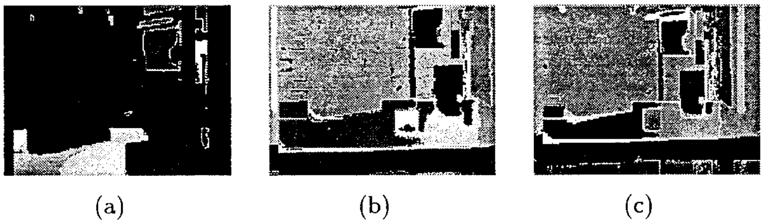


Figure 5.11: Samples of segmentation performance of the PHOENIX algorithm with default parameters on indoor color images (Figures, 5.7(a), 5.7(b) and 5.7(c), respectively).

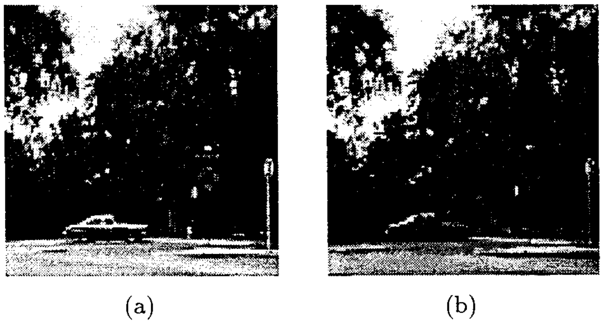


Figure 5.12: Samples of outdoor color images with varying environmental conditions. (a): Frame 2; (b): Frame 7.

Finally, Figure 5.16 shows the samples of performance of PHOENIX with default parameters on the outdoor color images shown in Figure 5.12. Note that these segmentation results are totally unacceptable.

5.5 Conclusions and Future Work

Our investigation into reinforcement learning-based closed-loop model-based object recognition shows that a robust and adaptive system can be developed that automatically de-

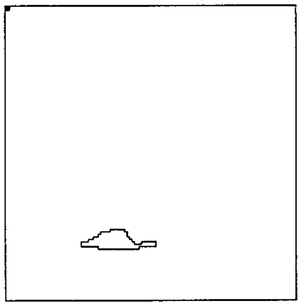


Figure 5.13: Polygonal approximation of the car used in the matching algorithm.

Table 5.2: Changes of parameter values during training.

Iteration	Hsmooth	Maxmin	Splitmin	Height
20	53	135	55	58
30	17	142	39	42
40	21	105	43	24
50	1	165	51	42
60	1	135	19	62
74	1	300	55	64

termines the criteria for segmentation of the input images and selects useful features which result in a system with high recognition accuracy when applied to new unseen images.

The key contribution of the chapter is the general framework for the usage of reinforcement learning in a closed-loop model-based object recognition system. Future research will address extensions for enlarging the scope of the approach in a multi-level object recognition system for practical applications.

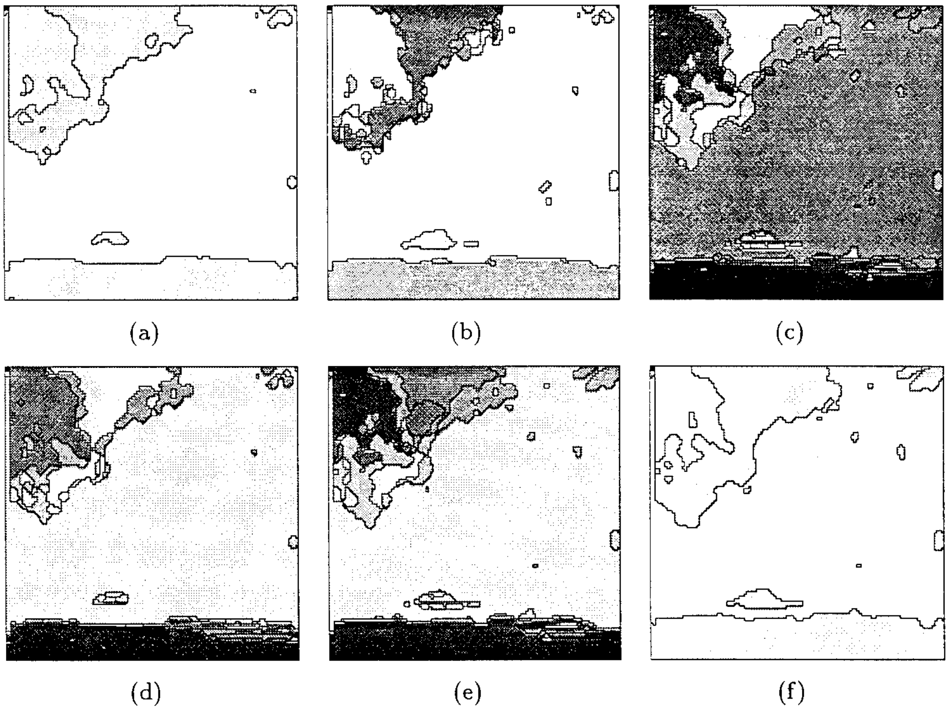
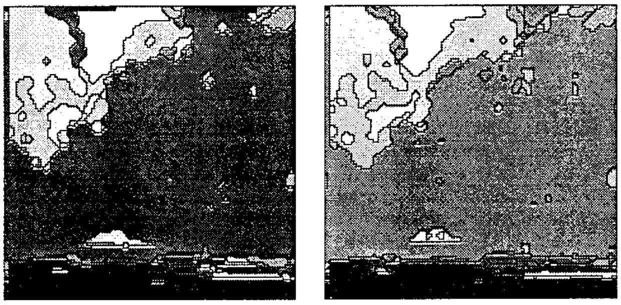


Figure 5.14: Sequence of segmentations of the first frame during training.

APPENDIX A: The Phoenix Segmentation Algorithm

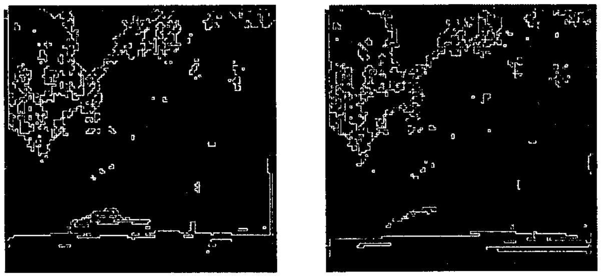
The Phoenix image segmentation algorithm is based on a recursive region splitting technique [60]. It uses information from the histograms of the red, green, and blue image components to split regions in the image into smaller sub-regions on the basis of a peak/valley analysis of each histogram. An input image typically consists of red, green, and blue image planes, although monochrome images, texture planes, and other pixel-oriented data may also be used. Each plane is called a feature or feature plane.

Figure 5.17 shows a conceptual description of the Phoenix segmentation process. It begins with the entire image as a single region. It then fetches this region and attempts to segment it using histogram and spatial analyses. If it succeeds, the program fetches each of the new regions in turn and attempts to segment them. The process terminates when no region can be further segmented.



(a) (b)

Figure 5.15: Segmentation performance of the PHOENIX algorithm on two testing images (frames 2 and 4) with learned parameters obtained after training.



(a) (b)

Figure 5.16: Samples of segmentation performance of the PHOENIX algorithm with default parameters on the two outdoor color images shown in Figure 5.12.

The histogram analysis phase computes a histogram for each feature plane, analyzes it and and selects thresholds or histogram cutpoints which are likely to identify significant homogeneous regions in the image. A set of thresholds for one feature is called an interval set. During the analysis, a histogram is first smoothed with an unweighted window average, where the window width is *hsmooth.* It is then broken into intervals such that each contains a peak and two "shoulders." A series of heuristics is applied to eliminate noise peaks. When an interval is removed, it is merged with the neighbor sharing the higher of its two

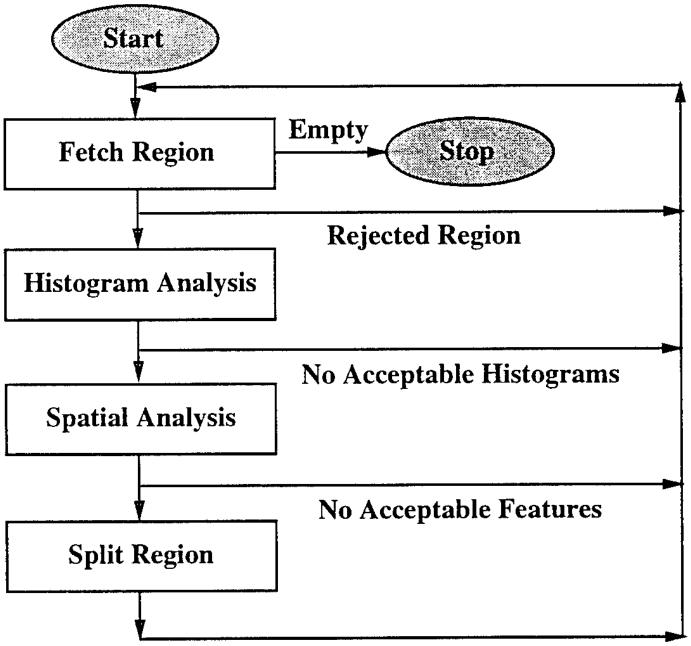


Figure 5.17: Conceptual diagram of the Phoenix segmentation algorithm.

shoulders. *Splitmin* is the minimum area for a region to be automatically considered for splitting.

Two tests determine if an interval should be retained. First, the ratio of peak height to the height of its higher shoulder must be greater than or equal to the *maxmin* threshold. Second, the interval area must be larger than an absolute threshold and the relative area, percent of the total histogram area. The second highest peak can now be found, and peaks lower than the *height* percent of this peak are merged. The lowest valley is then determined, and any interval whose right shoulder is higher than *absmin* (Phoenix's parameter) times this valley is merged with its right neighbor. Finally, only *intsmax* (Phoenix's parameter) intervals are retained by repeatedly merging intervals with low peak-to-shoulder ratio.

The spatial analysis selects the most promising interval sets, thresholds the corresponding feature planes, and extracts connected components for spatial evaluation. The feature and the interval set providing the best segmentation (the least noise area) are accepted as the segmentation feature and the thresholds.

The histogram cutpoints are now applied to the feature plane as intensity thresholds and connected components are extracted. After each feature has been evaluated, the one producing the least total noise area is accepted as the segmentation feature. If no suitable feature is found, the original region is declared terminal. Otherwise the valid patches, merged with the moise patches, are converted to new regions and added to the segmentation record. In either case, a new segmentation pass is scheduled. For additional details, see [60].

APPENDIX B: The Cluster-Structure Algorithm for Matching

The cluster-structure algorithm can be divided into the following main steps:

1. Determine Disparity Matrix
2. Initial Clustering
3. Sequencing
4. Final Clustering
5. Transform Computation

The algorithm first computes the disparity matrix. It determines the segment length of each line and the angles between successive lines from the set of vertices for the model and the image input to the program. At this point, every segment in the model will be compared against every segment in the image. If segment lengths and successor angles are compatible, the algorithm computes the rotational and translational disparity between pairs of segments. These values are stored in the disparity matrix and are indexed by the segment numbers in the model and the image. The algorithm continues until all segments have been compared. It then computes the range of rotational and translational values present in the matrix, and normalizes them over their appropriate range.

The initial clustering determines clusters from the normalized values in the disparity matrix. At each step, the program clusters all of the samples, recomputes the new cluster centers, and continues until none of the cluster centers change their positions. The program then selects the cluster having the largest number of samples. Also selected are the clusters which are within 20% of the largest one. Each cluster is considered separately and the final transform comes from the cluster that yields the highest confidence level.

The sequencing step uses the samples in the current cluster to find all sequences in the samples. This provides the critical structural information. Samples which are not placed in any sequence are discarded. The program also removes sequences that have a segment count of less than three (three segments comprise the basic local shape structure). It then computes the rotational and translation averages of each sequence that has been located.

Using the sequences and the sequence averages, the final clustering step clusters these values to find those sequences that lead to the same rotational and translational results. This is achieved by using the iterative technique of clustering, evaluating, clustering, etc. The program then selects the cluster that contains the largest number of sequences and passes this cluster to the final step.

The final step of the algorithm computes the confidence level of the transformation determined by each cluster. The cluster having the highest confidence level is selected as the final transformation cluster. It assembles the set of matched segments in the sequences in this cluster. The final output of the program is the rotation and the vertical and horizontal translation necessary to locate the model within the image. The program also produces a confidence level indicating the likelihood that the final matching is correct. For further details, see [19].

Chapter 6

Delayed Reinforcement Learning for Closed-Loop Object Recognition

Object recognition is a multi-level process requiring a sequence of algorithms at low, intermediate and high levels. Generally, such systems are open loop with no feedback between levels and assuring their robustness is a key challenge in computer vision research. A robust closed-loop system based on "delayed" reinforcement learning is introduced in this paper. The parameters of a multi-level system employed for model-based recognition are learned. The method improves recognition results over time by using the output at the highest level as feedback for the learning system. Appropriate credit in the form of rewards and penalties are assigned to the sequence of algorithms used for object recognition by the learning system. The method is experimentally validated by learning the parameters of image segmentation and feature extraction and thereby recognizing 2-D objects. The approach systematically controls feedback in a multi-level vision system and provides a solution to a long-standing problem in the field of computer vision.

6.1 Introduction

Most vision systems use a sequence of algorithms that operate at various levels of abstraction to perform a given task, such as object recognition. In earlier work that combines learning and vision [25, 78, 35], the inherent multi-level nature of vision systems has not been addressed adequately. In this chapter an approach that takes the output of the final level and uses it as a feedback in a reinforcement learning framework to influence the performance of the lower levels of vision algorithms is proposed. The overall system performance is improved over time with this method.

The improvement is possible via learning because vision systems are usually based on models of the physical world. For example, the process of creating the two-dimensional image from the three-dimensional world is usually modeled as a perspective projection. In machine learning terminology vision systems are said to exhibit a "bias." In other words, vision systems do not model some random phenomenon but are "biased" towards modeling the orderly physical world. Since it can be shown that learning is effective only in the presence of bias [65] it is possible to design a biased multi-level model-based object recognition system that improves its own performance over time.

The key to the improvement of a multi-level system over time is the automatic adjustment of parameters of various algorithms used in the system since the content of the three-dimensional scene and the imaging conditions are not known *a priori*. Currently, in most complex vision systems the designer manually adjusts parameters of the algorithms to some "default" values that are to be applied subsequently by users. However, the designer cannot anticipate for all possible inputs to the algorithms. The simultaneous adjustment of even a few system parameters is time-consuming and difficult and has yet to be solved satisfactorily for multi-level systems. The original contribution of this work is to provide an approach based on "delayed" reinforcement learning to control parameters in a multi-level object recognition system. A theoretical model is provided and its efficacy is validated on a moderately complex system. In contrast, the substantial body of work on system parameter estimation has not taken advantage of the power and flexibility of machine learning methods for multi-level vision systems.

With the above preliminaries consider the problem of model-based object recognition. Given the model of an object, the problem is to recognize it when it is located anywhere in an image. Figure 6.1 illustrates a typical approach to solve the problem. It segments the image at the first level, then extracts and selects appropriate features from the segmented image at the second level, and finally matches the selected features to the model. The segmentation and feature extraction modules use default parameters set by the system designer. However, the approach is inadequate for real-world applications because default parameters of segmentation and feature extraction often lead to large errors in recognition.

If it is assumed that the model matching produces a confidence measure indicating the closeness of the selected features to the model, then it is natural to use this confidence as feedback to influence the system's performance for segmentation and feature extraction.

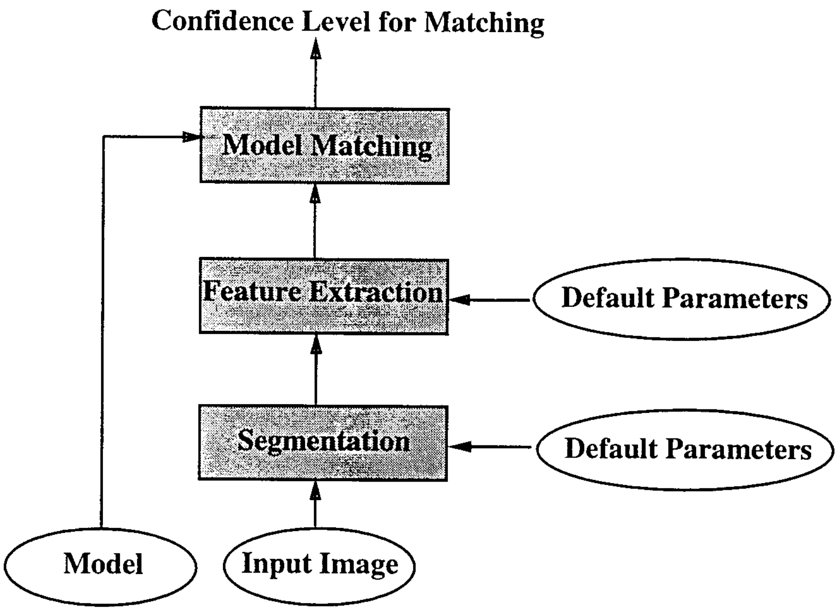


Figure 6.1: Conventional multi-level system for object recognition.

Figure 6.2 shows a closed-loop reinforcement learning-based system to achieve this goal. Reinforcement learning uses rewards and penalties based on the confidence to iteratively improve the performance of a system over time.

6.2 Reinforcement Learning System for Object Recognition

In the multi-level system for object recognition described in Figure 6.2 there are unknown parameters for both the segmentation and feature extraction modules. The segmentation[1] module is based on the "Phoenix" algorithm [60]. Phoenix uses region splitting based on histograms of color features and is critically dependent on system parameters "HSMOOTH" and "MAXMIN." HSMOOTH is the width of the histogram smoothing window with the smoothing performed by a uniformly weighted moving average technique. MAXMIN is the peak-to-valley height ratio threshold. Any interval whose peak height to shoulder ratio is

[1] For additional details on segmentation techniques see [50, 16].

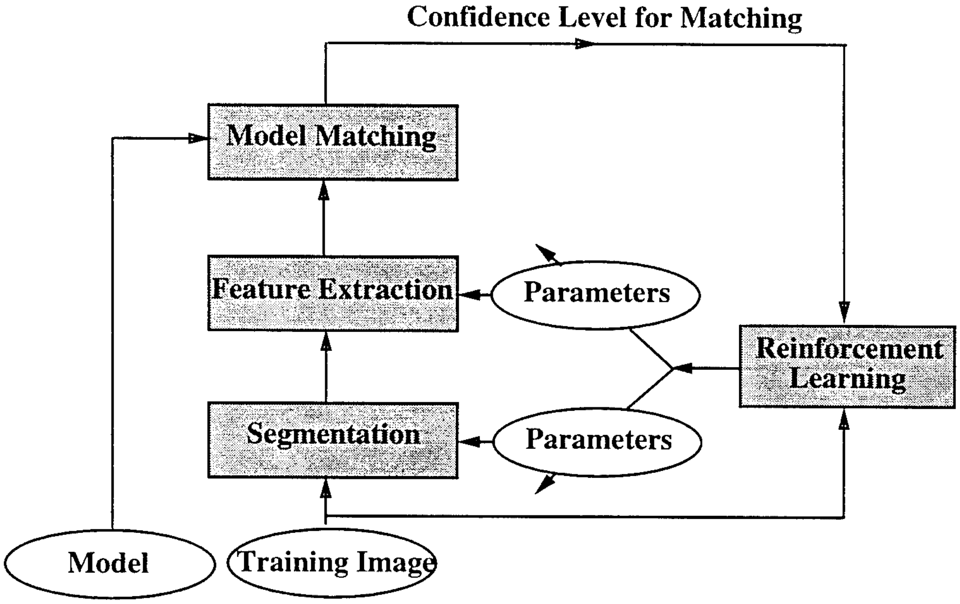


Figure 6.2: Reinforcement learning-based multi-level system for object recognition.

less than this threshold is merged with the neighbor on the side of the higher shoulder. The feature extraction module finds polygon approximation tokens for each of the regions obtained after image segmentation. The polygon approximation is obtained using a split and merge technique dependent on a parameter named "SMOOTH." SMOOTH is a quantitative measure of the smoothness of the polygonal approximation [19][2] . The model matching algorithm topologically compares a stored 2-D model to the token set output of the feature extraction module [19] [3]. It computes a real number that indicates the confidence level of the matching process.

It can be seen that the parameters HSMOOTH and MAXMIN are at the first level of the system and the parameter SMOOTH is at the second level. Reinforcement learning is used to adjust the parameters at both the first and second level. In reinforcement learning, at each time step a system is given not just perceptual inputs but also a numerical reward or penalty, called reinforcement. The reinforcement is a function of the output of the system

[2] HSMOOTH can take values ranging from 1 to 63, MAXMIN can take values ranging from 100 to 471 and SMOOTH can take values ranging from 6 to 16.

[3] For additional details on object recognition see [74, 27].

corresponding to the inputs at the previous time step. The goal of the system is to choose actions that maximize the sum of reinforcements (possibly discounted) over time [105].

One complication to reinforcement learning is the timing of reinforcement. In simple tasks, the system receives, after each decision, reinforcement indicating the goodness of that decision [79]. However, in most complex tasks reinforcement is often temporally delayed because immediate reinforcement regarding the value of a decision is unavailable. For example, in the object recognition system, the goodness of segmentation and feature extraction is not known until matching has been done.

An effective approach for delayed reinforcement learning is "temporal difference" (TD) learning [105]. In such learning systems, a "state" is a unique representation of all previous inputs to a system. The value of a state is regressed towards the weighted average of the values of its successors, where the weightings reflect the conditional probabilities of the successors, instead of the final outcome of reinforcement.

Let i be an input image to the segmentation module and a be an instance of segmentation parameters. Let $R(i,a)$ be the average immediate reinforcement for taking action a in input state i; $\gamma : 0 \leq \gamma \leq 1$ is a discount factor; j is the next input state resulting from taking action a in i; $p_{ij}(a)$ is the probability of going from state i to state j with action a; and $V(j) = max_b Q(j,b)$.

Then according to the Q-learning method $Q(i,a)$ measures how good the instance a is when applied to image i and is given by:

$$Q(i,a) = R(i,a) + \gamma \sum_j p_{ij}(a) V(j) \tag{6.1}$$

Q-learning works by updating the estimate of $Q(i,a)$ so that equation (6.1), with estimated values substituted for the unknown actual values, comes to be more nearly satisfied for each (i,a) encountered. If $R(i,a)$ of equation (6.1) is the expected value of r (reinforcement or confidence value) then this is done using the TD error:

$$r + \gamma V(j) - Q(i,a).$$

The particular reinforcement learning algorithm employed in the approach presented in this chapter is the Q(λ) learning algorithm [79, 107] which is a generalization of the Q-learning algorithm (details are given in the Appendix). It not only speeds up the learning but also allows for a non-Markovian environment encountered in vision applications. In terms of the example just described, this simply means that the value of $Q(i,a)$ will be corrected to look more like the value of the segmented image, which will in turn be estimated according to the matching confidence.

```
1. Initialize $Q$ function
2. LOOP:
   • For each image $i$ in the training set do
      (a) Image $i$, is segmented with current segmentation parameters $a = (a_1, a_2, \cdots, a_n)$; $i_s$ is the resulting segmented image.
      (b) Compute TD error: $V(i_s) - Q(i, a)$ and update $Q(i, a)$ according to the Q($\lambda$) learning algorithm.
      (c) Perform feature extraction with current values of feature extraction parameters $b = (b_1, b_2, \cdots, b_n)$, from the segmented image $i_s$.
      (d) Compute the matching of each extracted feature set against stored model and return the highest confidence level .
      (e) Compute TD error: $r + V(ABS) - Q(i_s, b)$ and update $Q(I_s, b)$ and $Q(i, a)$ according to the Q($\lambda$) learning algorithm. ($ABS$ is the absorbing state).
3. UNTIL terminating condition
```

Figure 6.3: Main steps of the delayed reinforcement learning algorithm for parameter adjustment for segmentation and feature extraction.

Figure 6.3 shows the main steps of the algorithm described conceptually in Figure 6.2. The algorithm terminates when either the number of iterations has exceeded a prespecified value or the recognition confidence level has reached a given threshold. Note that in general there can be multiple objects in the images.

6.3 Experimental Validation

There are several representation schemes for the Q-function in the reinforcement learning paradigm. Since the goal here is to isolate the effect of learning for multi-level recognition

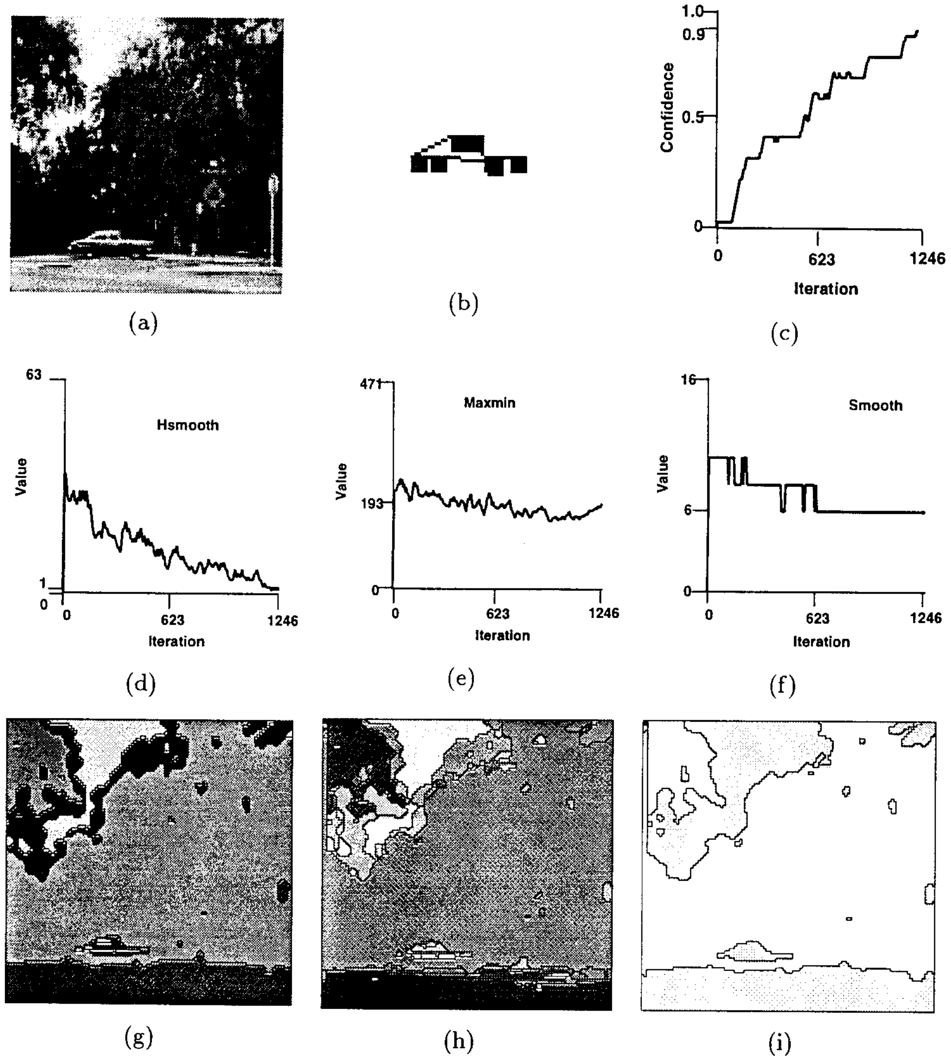


Figure 6.4: Experimental results for the training phase of an outdoor image.

a look-up table based representation suffices. The two dimensions of the look-up table are the following: (1) segmented or feature-extracted image, (2) action represented by a particular combination of system parameters. The focus of the experiments is to demonstrate the feasibility of using learning for multi-level recognition and not to address the issue of

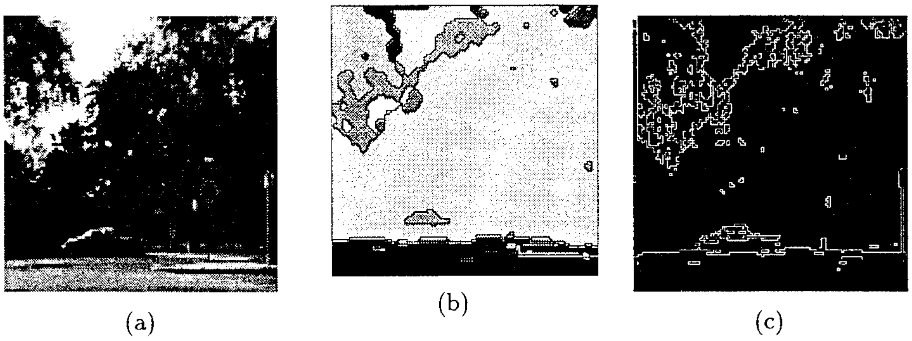


Figure 6.5: Experimental results in the testing phase on an outdoor image. (a) unknown image (b) segmentation with learned parameters (c) segmentation with default parameters.

generalization.

Figure 6.4 shows the results of the training phase of the system. Figure 6.4(a) shows a sample training image of a car on a road. The resolution of the image is 120 × 120 pixels. It should be noted that although the image is in color for publication purposes it is being shown in grayscale. Figure 6.4(b) shows the given 2-D model of the car located in Figure 6.4(a). The dark squares in Figure 6.4(b) correspond to labels of the vertices in the polygonal approximation of the car.

Figures 6.4(c), 6.4(d), 6.4(e), and 6.4(f) show how the confidence, HSMOOTH, MAXMIN and SMOOTH change over time, respectively. It should be noted that over time the confidence shown in Figure 6.4(c) increases. At the end of the training phase the confidence of the match is over 0.9 on a scale which varies between 0 and 1. For the purposes of our system an object is recognized if the confidence of matching is greater than 0.75. Furthermore, it should also be noted that the learned values of HSMOOTH, MAXMIN, and SMOOTH are considerably different from their starting values.

To illustrate the results further, Figures 6.4(g), 6.4(h), 6.4(i) show how the segmentation improves over time during the training phase. Figure 6.4(g) depicts the segmentation of the training image before applying the learning algorithm. Figure 6.4(h) depicts the segmentation after half the ιotal time for training has elapsed. Figure 6.4(i) depicts the segmentation at the end of the training phase. It can be seen that the results improve considerably.

Figure 6.5 shows the results in the testing phase on an outdoor image provided to the trained closed-loop object recognition system. Figure 6.5(a) shows an input image in which

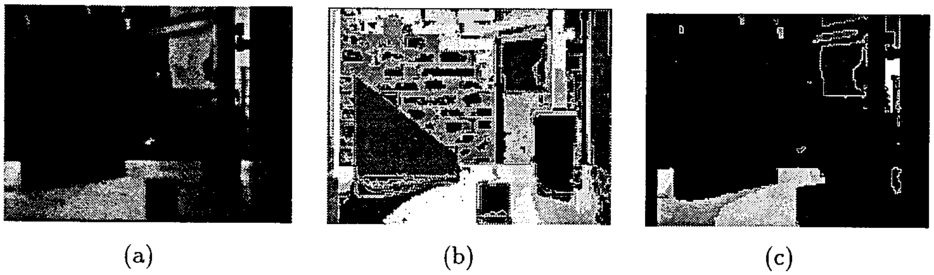


Figure 6.6: Experimental results in the testing phase on an indoor image. (a) unknown image (b) segmentation with learned parameters (c) segmentation with default parameters.

the car of Figure 6.4(b) must be identified. It can be seen that the lighting conditions in the outdoor image of the testing phase is significantly different from the training image. The image is taken at a different time from Figure 6.4(a). Observe that there are significant changes between the cars shown in Figures 6.4(a) and 6.5(a). Figures 6.5(b) and 6.5(c) show the segmentation obtained by using the parameters obtained from the training phase, and the segmentation obtained by using default parameters, respectively. It should be noted that when default parameters are used the car is broken up into many small blobs. The confidence of model matching was obtained as 0.88. It should also be noted that delayed reinforcement learning has been used only in the training phase.

Figure 6.6 shows results in the testing phase of another image except that now it is of an indoor scene. For brevity the results for training are not shown for this scenario. Figure 6.6(a), 6.6(b), and 6.6(c) show the indoor image, the segmentation with learned parameters and the segmentation with default parameters respectively. The large triangular shaped object (wedge) is the object of interest. The confidence of model matching was 0.85.

It should be noted that until the final recognition outcome is determined the effectiveness of the segmentation and feature extraction modules cannot be ascertained. Experimental results show that a robust and adaptive system can be developed that determines autonomously the criteria for segmentation and feature extraction to achieve a high accuracy for recognition on new images.

6.4 Conclusions and Future Work

To summarize it can be stated that a multi-level approach to object recognition in a "delayed" reinforcement learning paradigm was described in this chapter. The reinforcement learning algorithm used rewards and penalties at successive levels of a closed-loop model-based object recognition system to iteratively improve the system parameters. A system was built to exemplify the efficacy of the approach. After feature extraction from the segmented image, the system computed the confidence of matching between the features and the model. Reinforcement learning used the confidence to adjust the segmentation and feature extraction parameters in such a way that the confidence of matching improved significantly over time. Simple objects in indoor and outdoor scenes were recognized in about a thousand time steps even when the reinforcement learning algorithm started with random values of the parameters. In contrast, "default" parameters of the system gave a poor confidence of matching.

If vision systems could be designed in one-level as a single black box the "simple" reinforcement paradigm would have sufficed. Earlier work on one-level systems used a team of stochastic semi-linear units for learning image segmentation parameters [78]. However, in reality both open and closed-loop systems have multiple levels with parameters that need to be adjusted at each level. Delayed reinforcement learning allowed an elegant and effective solution to the problem of object recognition in multi-level systems. The system presented here included the recognition component as part of the evaluation functions for learning in a systematic way. The emphasis here was not so much in simple mixtures of learning and computer vision, but rather in the principled integration of the two fields at the algorithmic level. The key contribution is the general framework for the usage of delayed reinforcement learning in a multi-level vision system. Future research will address extensions for enlarging the scope of the approach to encompass problems in active vision where reinforcement learning could be extremely useful. Furthermore, incorporatation of more efficient representations could facilitate the study of generalization issues pertaining to the system.

APPENDIX A: Q(λ) Learning

One set of methods for determining an optimal policy is given by the theory of dynamic programming. These methods entail first determining the "optimal state-value function", V, which assigns to each state the expected total discounted reward obtained when an optimal policy is followed starting in that state. As in [107], a closely related function can be defined that assigns to each state-action pair a value measuring the expected total discounted reward obtained when the given action is taken in the given state and the optimal policy is followed thereafter. That is, using the notation that x denotes the current state, a the current action, r the resulting immediate reward, and y the resulting next state,

$$\begin{aligned} Q(x,a) &= E\{r+\gamma V(y)|x,a\} \\ &= R(x,a)+\gamma\sum_y P_{xy}(a)V(y), \end{aligned} \tag{6.2}$$

where $R(x,a) = E\{r|x,a\}$, $V(x) = \max_a Q(x,a)$, and $P_{xy}(a)$ is the probability of making a state transition from x to y as a result of applying action a.

Note that once we have this Q-function it is straightforward to determine the optimal policy. For any state x the optimal action is simply $\arg\max_a Q(x,a)$.

The *Q-learning* algorithm is based on maintaining an estimate $\hat{Q}$ of the Q-function and updating it so that equation (6.2), with estimated values substituted for the unknown actual values, comes to be more nearly satisfied for each state-action pair encountered. More precisely, the algorithm is as follows: At each transition from one time step to the next, the learning system observes the current state x, takes action a, receives immediate reward r, and observes the next state y. Assuming a tabular representation of these estimates, $\hat{Q}(x,a)$ is left unchanged for all state-action pairs not equal to (x,a) and

$$\hat{Q}(x,a) \leftarrow \hat{Q}(x,a)+\alpha\left[r+\gamma\hat{V}(y)-\hat{Q}(x,a)\right], \tag{6.3}$$

where $\alpha \in (0,1]$ is a learning rate parameter and $\hat{V}(y) = \max_b \hat{Q}(y,b)$. An estimate of the optimal action at any state x is obtained in the obvious way as $\arg\max_a \hat{Q}(x,a)$.

This algorithm is an example of the temporal difference method because the quantity $r+\gamma\hat{V}(y)-\hat{Q}(x,a)$ can be interpreted as the difference between two successive predictions of an appropriate expected total discounted reward. The general effect of such algorithms is to correct earlier predictions to more closely match later ones.

The advantage of the Q-learning algorithm is that when combined with sufficient exploration it can be guaranteed to eventually converge to an optimal policy [107]. The disadvantages, however, are that it is very slow to converge and may work poorly in problem domains which are non-Markovian. To overcome these weaknesses, Peng and Williams

```
1. $\hat{Q}(x,a) = 0$ and $Tr(x,a) = 0$ for all $x$ and $a$
2. Do Forever:
    (a) $x_t \leftarrow$ the current state
    (b) Choose an action $a_t$ that maximizes $\hat{Q}(x_t, a)$ over all $a$
    (c) Carry out action $a_t$ in the world. Let the short term reward be $r_t$,
        and the new state be $x_{t+1}$
    (d) $e'_t = r_t + \gamma \hat{V}_t(x_{t+1}) - \hat{Q}_t(x_t, a_t)$
    (e) $e_t = r_t + \gamma \hat{V}_t(x_{t+1}) - \hat{V}_t(x_t)$
    (f) For each state-action pair $(x, a)$ do
        • $Tr(x,a) = \gamma \lambda Tr(x,a)$
        • $\hat{Q}_{t+1}(x,a) =$
          $\hat{Q}_t(x,a) + \alpha Tr(x,a) e_t$
    (g) $\hat{Q}_{t+1}(x_t, a_t) = \hat{Q}_{t+1}(x_t, a_t) + \alpha e'_t$
    (h) $Tr(x_t, a_t) = Tr(x_t, a_t) + 1$
```

Figure 6.7: The Q(λ)-learning algorithm used in our approach.

[79] have introduced the Q(λ) learning algorithm in which the current prediction error is used to correct previously experienced state-action pairs in addition to the current one. More formally, the following form of evaluation function estimators [105] is used in Q(λ) learning:

$$\mathbf{r}_t^\lambda = r_t + \gamma(1-\lambda)\hat{V}_t^\pi(x_{t+1}) + \gamma\lambda\mathbf{r}_{t+1}^\lambda \tag{6.4}$$

Equation (6.4) is called TD(λ) estimators [105]. Then the Q(λ) learning algorithm can be derived in Figure 6.7, where $Tr(x,a)$ is the "activity" trace of state-action pair (x,a), corresponding to the "eligibility" trace as described in literature. It is worth noting when $\lambda = 0$ then Q(λ) learning reduces to Q-learning.

Chapter 7

Context Reinforced Background Modeling

Automatic Target Detection and Recognition (ATD/R) may be considered as an extension of object recognition problem from its original successful domain of simple objects in the *block world* to a more difficult new domain of complex objects embedded in natural environment. Such an extension has seen some new challenges that do not exist in the block world. First, the target may appear in front of many different outdoor backgrounds, e.g. the target may be seen in the desert areas of Africa or it can also be seen in the forests of North Europe. The appearance of the background is totally out of our control, and we have no means to adjust the background to get a high contrast between the target and the background. Secondly, there are various practical restrictions that prevent us from taking a high quality image about a specific scene. If the ATD/R system had to get close enough to an enemy's tank in order to recognize it, it would be very likely that the ATD/R system is destroyed by the enemy before any recognition result is obtained. Finally, many covering techniques have been developed to hide the identity of the target. As a result, most input images to a ATD/R system are of low resolution and high clutter, and the targets may also be partially occluded. When traditional object-model-based object recognition systems are required to handle these low resolution images, they are not successful. Two indexes are important to ATD/R performance, one is the *probability of detection* and the other is the *probability of false alarm.* With traditional object recognition approaches, whenever we improve one of the index we usually sacrifice the other. Since the output of an ATD/R system is normally used to control the weapon system, both performance indexes need to be maintained at a reasonable level.

From our point of view, part of the reason why it is difficult to apply most existing object recognition approaches to ATD/R is because the background of the concerned object is generally ignored by these approaches, and the recognition processes normally begin with a segmentation stage which is nothing but "background rejection." In most indoor environment where the background of the concerned object can be selected or controlled to get a high object-background contrast, this task is simple. But for ATD/R in cluttered environments, where targets become more mixed with their backgrounds, high quality early segmentation becomes very difficult.

In order to cope with this problem, we propose here a new strategy called *Background Model Aided Target Detection and Recognition* (BMATD/R). The main idea of this strategy is to maintain a high probability of detection while reducing the number of false alarms by involving explicit *background models* into ATD/R processes. The practice of modeling the background has long existed in the field of audio signal processing, synthetic aperture radar (SAR) signal analysis, infrared detection techniques and image processing, where it is usually referred to as noise model or clutter model. A common practice in building these models is to use existing stochastic process models to set up a framework for the behavior of the noise or clutter, and data from real signals is used to decide the value of free parameters. Such an approach is seldom seen for ATD/R, because no existing stochastic model has a solid proof of its validity (either theoretical or experimental) for describing the appearance of natural backgrounds in an image. As a result, most of the effort has been devoted to finding more sophisticated and robust target modeling techniques and developing new matching algorithms that can tolerate more feature distortions. Although efforts in this direction have led to some encouraging results, we believe background modeling and recognition can make this progress quicker by putting it onto two wheels instead of one.

7.1 Representation of A Background Model Bank (BMB) Member Using A Self-organizing Map

In our previous report, we pointed out that high cluttered sensory data had made it hard to extract perfect object features from the input, which are crucial for a conventional object recognition system to have a good performance. As a result, more and more sophisticated feature sets are introduced to compensate the weakness of any single metric. And the dimension of the feature vectors has grown to such an extent that it is already impossible for human beings to visualize and understand the train of thought of the undergoing recognition process. While this situation may be tolerable in an unsupervised learning environment, it would be very hard to conduct a high quality supervised learning with the supervisor himself being blind. Based on such a concern, we will try to avoid using high dimension feature sets during the building up of our background models. Rather than constructing a sophisticated model for a certain natural background based on a very complicated feature set, we would try to investigate independently each available group of metrics that have closely related physical meaning, and build one model based on each such metric group. Thus for a certain background we will have a bank of simple models, each model is called a **member** of the bank, and it views the under-investigated background based on its own "theory". Such a **background model bank** (BMB) is superior, in the following aspects, to the all-in-one model that is widely used in current Automatic Target Detection and Recognition (ATD/R) research:

- *The BMB is more suitable for supervised learning environment:* Since each member of a BMB is simple (i.e. based on very short feature vectors) it would be easy to visualize the modeling process and thus enable the supervisor to "see" the learning process, so he could select examples with proper difficulty to speed up the learning process.

- *The BMB makes it easier to involve new metrics into ATD/R systems:* Whenever a group of new metrics is found useful for modeling and recognition of the background, a new member can be created and inserted into the BMB. The all-in-one models would need much more work to increase the dimension of their feature vectors.

- *The BMB is more easily to be extended to multi-sensor based ATD/R systems:* Since each member of the BMB is investigated independently, involving a new sensory input is just to add a new member into the BMB.

- *The BMB will provide an efficient way to manage the long-term knowledge cumulation:* Based on the above two advantages, BMB approach can keep improve its performance by involving new sensory techniques and new image processing methods as they emerge. Such a property is very important because we can expect that a successful ATD/R system will need time to become mature.

Although many papers in the literature have used known statistical distributions in their analysis of natural clutters in IR images, there is no strong evidence that thermal natural clutters possess a certain statistical distribution [87]. Instead of artificially assigning a distribution model to background models, we construct our BMB from real images through a supervised learning process. Since reliable statistical models can only be obtained through analysis of a large population of samples, space and time complexities of algorithms become a major concern when selecting a learning scheme. In our approach, each BMB member is represented by a self-organizing map (SOM). By controlling the size of the SOM, we can easily control the space and time complexity of the learning process. Figure 7.1 shows the training process for a BMB member. A supervised SOM algorithm has been developed to

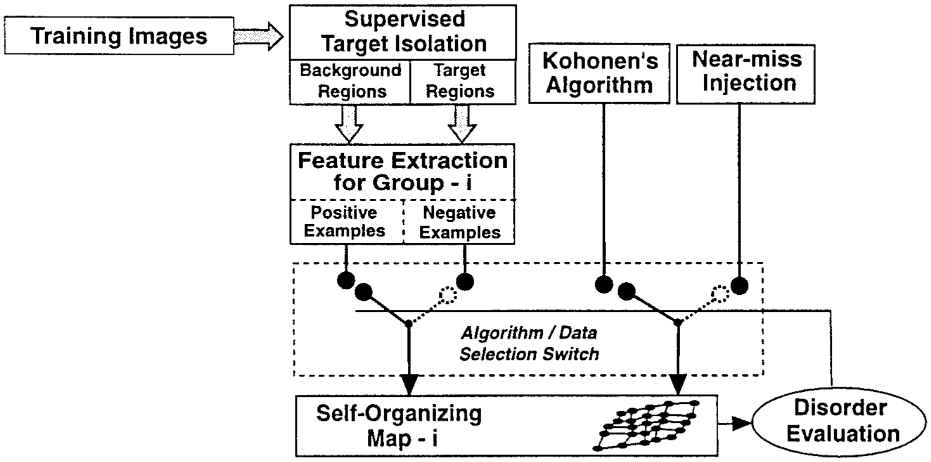


Figure 7.1: Building up a member of the Background Model Bank. The initial uniformly distributed self-organizing map (SOM) is trained first by using positive examples and Kohonen's algorithm. After a pre-selected number of iterations, a disorder index is computed. If the map has reached a certain degree of ordering, the algorithm/data selection switch is turned to the near-miss injection algorithm which uses negative examples to refine the trained SOM. To allow a BMB member to memorize its valuable past knowledges while it gains new experiences, the size of the SOM needs to be extensible. An incremental SOM algorithm allows us to achieve this.

accomplish learning for BMB members from both positive and negative training examples [89].

7.2 Conventional Self-organizing Maps

In Kohonen's SOM algorithm, neurons are arranged into an $N \times N$ array. After initialization of the weight vector $\mathbf{w}_i$ of each neuron i, the algorithm runs inside a loop which contains two operations:
(1) given a training feature vector $\mathbf{x}$, search is carried out for the winning neuron c which fulfills

$$||\mathbf{x} - \mathbf{w}_c|| = \min_i ||\mathbf{x} - \mathbf{w}_i||, \mathrm{i} = 1, 2, \cdots, N^2 \qquad (7.1)$$

(2) update the weight vectors of the winning neuron c and every neuron within a neighborhood of c according to

$$\mathbf{w}_i(t+1) = \begin{cases} \mathbf{w}_i(t) + \alpha(t)\,(\mathbf{x}(t) - \mathbf{w}_i(t)) & \text{for i} \in N_c \\ \mathbf{w}_i(t) & \text{otherwise} \end{cases} \tag{7.2}$$

Different strategies can be used to control the learning rate $\alpha(t)$ and to adjust the neighborhood N_c as training goes on. Both parameters should decay with time. In the above algorithm, normally the training process terminates when a pre-selected iteration number has been reached. The selection of this number is mainly based on experiments.

7.3 Supervised Self-organizing Maps

Learning From Positive Examples

The first step of the supervised self-organizing map algorithm is to use the Kohonen's algorithm to train the SOM by using positive training examples. By positive examples we mean pure background images that have no target embedded in them. During learning, a group of such images will be presented to the learning system for generating positive feature vectors for each feature group.

7.3.1 Disorder Index

When applying kohonen's algorithm to real world problems, people often find that it needs a lot of experiments to select a good set of parameters. The termination criterion is one of them. To make the learning process autonomous, i.e. without the need for humans intervention, a metric reflecting the SOM's ordering is needed so that the algorithm can determine how well the SOM has been trained, and thus determine whether it is time to terminate the learning process. In our research, we developed two metrics to describe the ordering of a SOM. The first one is based on the proved asymptotic convergence property of the SOM, and the second one is based on a direct analysis of the distortion of the SOM grid.

Disorder index 1

Since a properly trained SOM asymptotically converges to the distribution of training examples, the variation of the weight vectors with respect to a fixed number of training iterations will decrease asymptotically. So a measure based on this variation can be used as an index for the ordering of SOM. Let $d_{ms}(t)$ be the mean square distance between the training vectors and the weight vectors at discrete training time t, we have

$$d_{ms}(t) = \frac{1}{|S_T|} \sum_{\mathbf{x} \in S_T} \left(\frac{1}{|N_c|} \sum_{i \in N_c} ||\mathbf{x} - \mathbf{w}_i(t)||^2 \right) \tag{7.3}$$

where S_T denotes the training set and N_c is the set of neurons within the neighborhood of the winning neuron. The *Disorder Index* (*DOI*) can then be defined as

$$DOI = d_{ms}(t + k) - d_{ms}(t) \tag{7.4}$$

where k determines the length of the interval when DOI is evaluated. Recently a more sophisticated metric has been proposed for measuring the disorder of a SOM [68].

Disorder index 2

Because all our feature groups and their corresponding SOM's are 2 dimensional, we can directly analyze the distortion of the $2D$ neuron grid and use the result as the disorder index. To be judged as having been well ordered by this disorder index, a SOM must satisfy two conditions:

- All the extreme neurons must be *boundary neurons.*
- The ratio between the number of distorted grids and the number of grids of the SOM must be less than a preselected threshold.

Although we also need a preselected threshold to apply this disorder index, it is much more easier for a learning supervisor to select this threshold, compared with the total iteration threshold used in Kohonen's algorithm. So, the disorder index can be formulated as following:

$$DOI = \begin{cases} Th + 1 & \text{if } \{boundary\} \neq \{extrem\} \\ Nd/Nt & \text{if } \{boundary\} = \{extreme\} \end{cases} \tag{7.5}$$

where Th is the preselected threshold for this disorder index, Nd is the number of distorted grids, and Nt is the total number of grids in the SOM. $\{boundary\}$ is the set of boundary neurons and $\{extrem\}$ is the set of the extreme neurons .

7.3.2 Learning From Negative Examples

When *DOI* is below a pre-selected threshold the SOM is in a well ordered state, and a conventional SOM algorithm can terminate its learning process at this time. In our approach, at this time the learning process will go into the second stage — refining those ambiguous regions in the SOM by using the *near-miss injection* algorithm and negative examples. By ambiguous regions we mean regions where features of different classes (e.g. background and man-made target) overlap. The near-miss injection algorithm runs inside a loop which contains two steps:

(1) given a negative training vector $\mathbf{y}$, search is made for the "hitting" neuron h using equation 7.1.

(2) update the weight vectors according to

$$\mathbf{w}_i(t+1) = \begin{cases} \mathbf{w}_i(t) + \frac{\beta(t)\mathbf{u}}{(||\mathbf{y}-\mathbf{w}_i(t)||+1)^2} & \text{for i} \in N_h \\ \mathbf{w}_i(t) & \text{otherwise} \end{cases} \tag{7.6}$$

$$\mathbf{u} = \begin{cases} \frac{\mathbf{y}(t)-\mathbf{w}_i(t)}{||\mathbf{y}(t)-\mathbf{w}_i(t)||} & \text{if } ||\mathbf{y} - \mathbf{w}_i(t)|| \neq 0 \\ \frac{\mathbf{y}(t)-\bar{\mathbf{w}}_i(t)}{||\mathbf{y}(t)-\bar{\mathbf{w}}_i(t)||} & \text{if } ||\mathbf{y} - \mathbf{w}_i(t)|| = 0 \end{cases} \tag{7.7}$$

$$\bar{\mathbf{w}}_i = \frac{1}{8}\sum_j \mathbf{w}_j, \text{ neuron } j \in \text{4-neighbor of neuron } i \tag{7.8}$$

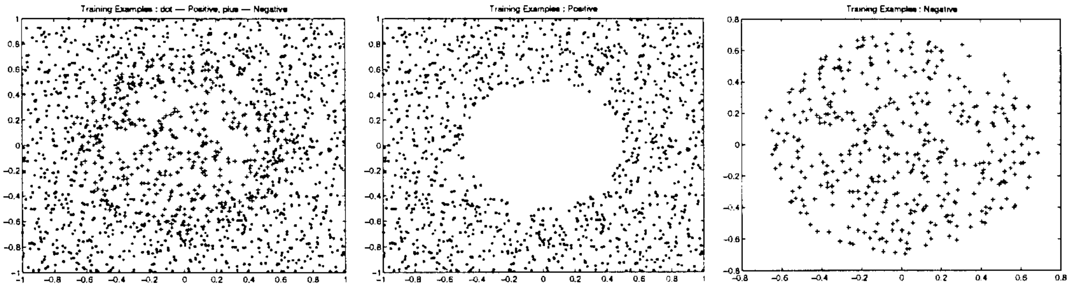
where $\beta(t)$ is the learning rate for this near-miss injection algorithm, it should decay with time, and we can use the same decay function used in Kohonen's algorithm to control β.

7.4 Experimental Results

In our experiment, we compared our supervised *SOM* algorithm, which uses both positive and negative examples during the training, with the conventional Kohonen's algorithm, first using synthetic testing data, then using real data — the feature values computed from real testing images.

7.4.1 Synthetic Data

Shown in Figure 7.2 are the synthetic data. Positive examples are evenly distributed over a 2 × 2 square area, with a hole in the center. Negative examples are evenly distributed



(a) (b) (c)

Figure 7.2: Synthetic data for testing SOM algorithm. (a) Positive examples overlapped with negative examples. (b) Positive examples. (c) Negative examples.

in a circular area with a radius of 0.7. The overlapping region is a ring with a width of 0.2. The size of the SOM used in the experiment is 7 × 7. Figure 7.3 shows the trained SOM by applying Kohonen's algorithm for 1000 epochs. Figure 7.4 shows distribution of $4 - Neighber$ the average distance both for all the positive examples and all the negative examples. With this distribution, we computed two classification thresholds, Thp and Thn, which are the mean $4 - Neighber$ average distance positive examples and negative examples. By using Thp and Thn, each training example (positive and negative) is classified using the trained SOM. Among total 795 positive examples, 171 were misclassified as negative, while among 192 negative examples, 45 were misclassified as positive.

Figure 7.5 and Figure 7.6 are the result by applying our supervised SOM algorithm for 1000 epochs. With the same number of positive and negative examples, the two misclassification measures are 114 and 43.

7.4.2 Real Data

Twenty FLIR images like those shown in Figure 7.7 were used to built the background model for target detection. Shown in Figure 7.8 is the distribution of the LSGE feature values extracted from these 20 training images. Both Kohonen's algorithm and our supervised SOM algorithm were used to generate a set of representatives from the training feature vectors.

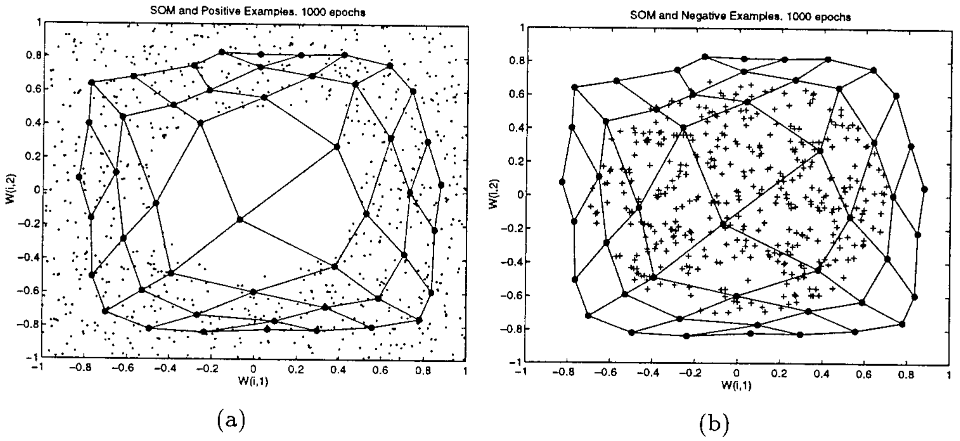


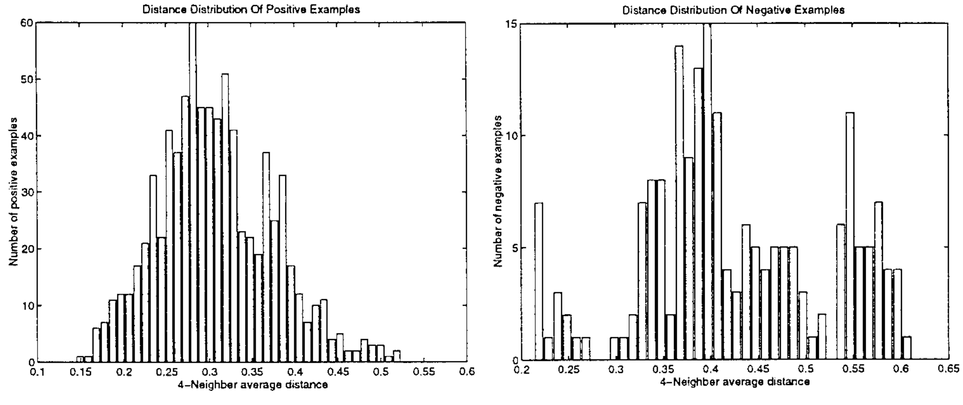
Figure 7.3: Trained SOM by applying Kohonen's algorithm for 1000 epochs. (a) SOM overlapped with positive examples. (b) SOM overlapped with negative examples.

The constructed SOM using Kohonen's algorithm from the absolute LSGE feature data is shown in Figure 7.9. The SOM corresponding to the relative LSGE feature data is shown in Figure 7.10. The mis-classification ratio is 6/239 and 12/35 for the absolute LSGE feature group. The ratio for the relative LSGE feature group is 4/239 and 15/35. The same procedure was repeated for the supervised SOM algorithm. The resulted SOM is shown in Figure 7.11 and Figure 7.12 for the two feature groups. The mis-classification ration is 4/239 6/35 for the absolute LSGE and 4/239 and 10/35 for the relative LSGE feature group.

7.5 Validity Scopes of The Background Models

7.5.1 The Role of Contextual Parameters

Automatic Target Detection and Recognition (ATD/R) is a challenging application for the general techniques developed by image processing and image understanding communities. This challenge is mainly due to the lack of control of the environment in a typical ATR mission. As a result, there are many variables that can affect the performance of an ATR



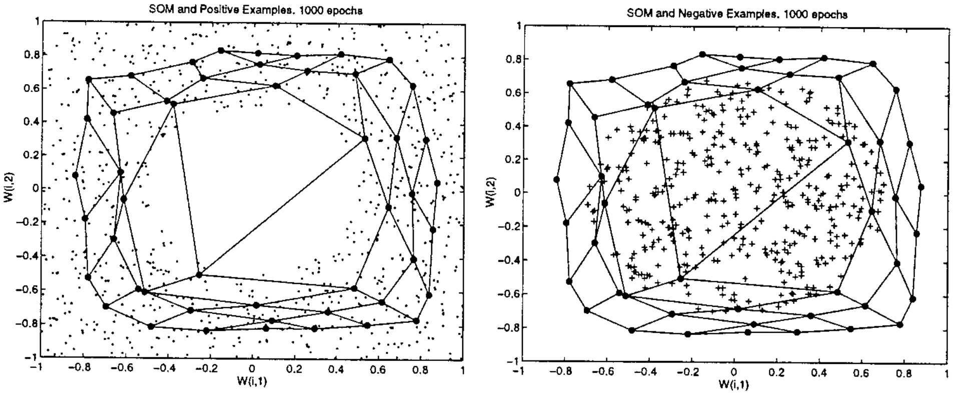
(a)

(b)

Figure 7.4: Distribution of $4 - Neighber$ average distance of (a) all positive examples. (b) all negative examples.

system. Sherman et al. [59] categorized 41 such variables into five classes — background parameters, target parameters, platform dynamics, atmospherics and sensor parameters. Because a target could appear on the same background under different contextual conditions, e.g. different time of the day, different air temperature, or being viewed with a different depression angle, when we build the BMB, we should also cover this variation of contextual conditions.

It can be imagined that different features may have different sensitivity to a certain contextual parameter, e.g. the mean and standard deviation of image gray values are more sensitive to the air temperature than the Gabor transform amplitude features which tend to find out the periodic pattern with in a local image region. To be practically applicable, a ATD system must be able to detect targets under different contextual conditions. One way to achieve this goal is to use learning technique to associate contextual parameters with the performance of each feature group. The rationality behind this association is that if a feature group can effectively detect man-made objects under a given contextual condition, it tends to be effective for images taken under similar contextual conditions. Since the human supervisor cannot provide any assistance to the ATD system in finding this association, except telling the system whether it is doing a good job with respect to a specific testing image, the most suitable learning scheme for this task is the *reinforcement learning* scheme.



(a)

(b)

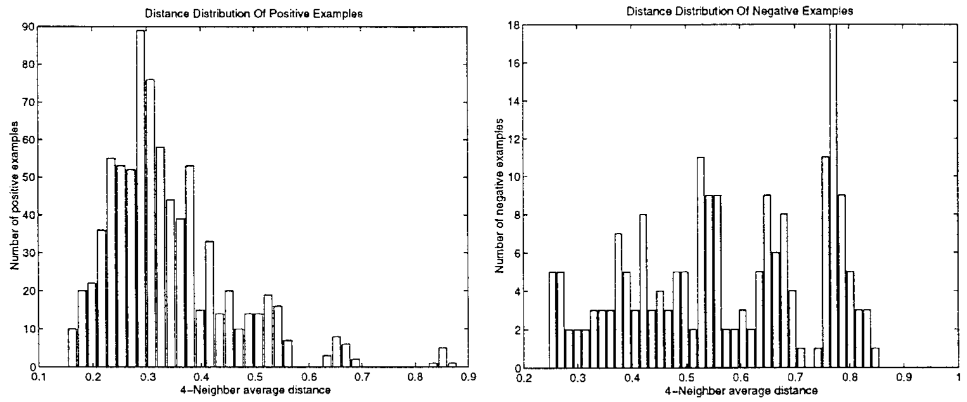
Figure 7.5: Trained SOM by applying supervised SOM algorithm for 1000 epochs. (a) SOM overlapped with positive examples. (b) SOM overlapped with negative examples.

7.5.2 Reinforcement Learning Using Contextual Parameters

If a feature group has a good performance under a certain contextual condition, its detection result deserves a heavy weight for all the similar contextual conditions. In another word, the context — performance relationship can be replaced by a context — weight relationship, which is more compliant for being integrated into a automatic learning system. To facilitate the discussion, we define the following terms which will be used to formulate the SRV based algorithm.

- *Contextual Parameter* (c) is a scalar that quantifies a specific aspect of a contextual condition, it can be defined over continuous or discrete values.
- *Contextual vector (C)* is a vector with each element being c^i, a *contextual parameter.*
- Weight Vector (W) is a real value vector with each element being w^i, the weight of a feature group.

Our learning problem can then be defined as following:



(a) (b)

Figure 7.6: Distribution of $4 - Neighber$ average distance of (a) all positive examples. (b) all negative examples.

Given a set of training images that cover the whole range of available contextual conditions, with the BMB having been built as a collection of SOM's, we would like to associate with each BMB member SOM^i a stochastic transform function T^i, such that

$$w^i = T^i(\mathbf{C})$$

.

T^i is stochastic because the C — W relationship can not be described by a deterministic function, there are always exceptional cases due to the high complexity of the real world.

7.5.3 The SRV Algorithm

The reinforcement learning algorithm we selected for learning the context — performance relationship is called *Stochastic Real Valued reinforcement learning (SRV)* algorithm, developed by Gullapalli [48, 47]. This algorithm allows the system to learn outputs that take on real values. Since the performance of a feature group is best described as a real number, normally from 0 to 1, with 1 representing the best performance, this SRV algorithm meets our requirement very well.

In SRV algorithm, T^i is implemented as a random number generator according to the normal distribution. The mean μ^i, and standard deviation σ^i are determined by two internal vectors, Φ^i and Ψ^i according to the following formula.

$$\mu_n = \Phi^T \cdot C_n \tag{7.9}$$

$$\sigma_n = 1 - \hat{r}_n = 1 - f(\Psi^T \cdot C) \tag{7.10}$$

where function $f(\cdot)$ often takes the form of

$$f(a) = \frac{1}{1 + e^{-a}} \tag{7.11}$$

Once the two parameters are available, the weight for the ith feature group can be computed by passing μ^i and σ^i to a random number generator:

$$w^i \sim N(\mu^i, \sigma^i)$$

.

So, in the learning system, the transform function T is actually "remembered" as two vectors, Φ and Ψ. Starting with random selected initial values, these two internal vectors learn to represent the C — W relationship by updating themselves according to the following formula.

$$\Phi_{n+1} = \Phi_n + \sigma_n(r_n - \hat{r}_n)(w_n - \mu_n)C_n \tag{7.12}$$

$$\Psi_{n+1} = \Psi_n + \rho_n(r_n - \hat{r}_n)C_n \tag{7.13}$$

where

$$r_n = g(P)$$

is the reinforcement provided by a critic function $g(\cdot)$ for the nth detection trial. Vector P is the detection result vector that can be used by the *critic* to judge the performance of the system after the detection trial.

7.5.4 Implementation Concerns

To utilize this complex learning scheme to solve the previously defined $C — W$ problem, we have to make several implementation decisions.

1. *selection of contextual parameters* : It is obvious that we cannot deal with 41 contextual parameters at the same time. One practical way is to select a subset from the available contextual parameters. In our implementation, we selected 4 parameters to form the contextual vector, they are

   - *t : Time of the day.*
   - *d : Depression angle.*
   - *s : Range to the target.*
   - *p : Air temperature.*

   In order to make the inner product of Equation 7.9 and Equation 7.10 meaningful, we used relative values of the contextual parameters in constructing the contextual vector. The relative value of d, for example, is $\frac{d}{d_{max}-d_{min}}$, where d_{max} and d_{min} are the maximum and minimum depression angle occured in the training images.

2. *the performance vector P* : Since all our features are region based features, given a testing image, the image is first divided into rectangular regions based on the *Range to the target* information. The detection result is a label map l that labels each region either as a background region or a target region. The easiest way to describe the performance of the detection is to compare l with the label map L given by the learning supervisor. Thus, the performance vector p can simply be $p = l - L$.

3. *selection of the critic function :* Since we are dealing with a two class classification problem, both l and L can be a bit vector. A simple metric for the detection performance is obtained by examine the number of bits being set to 1 in p,

   $$r_n = \sum_{i=1}^{Nb} p(i)/Nb \tag{7.14}$$

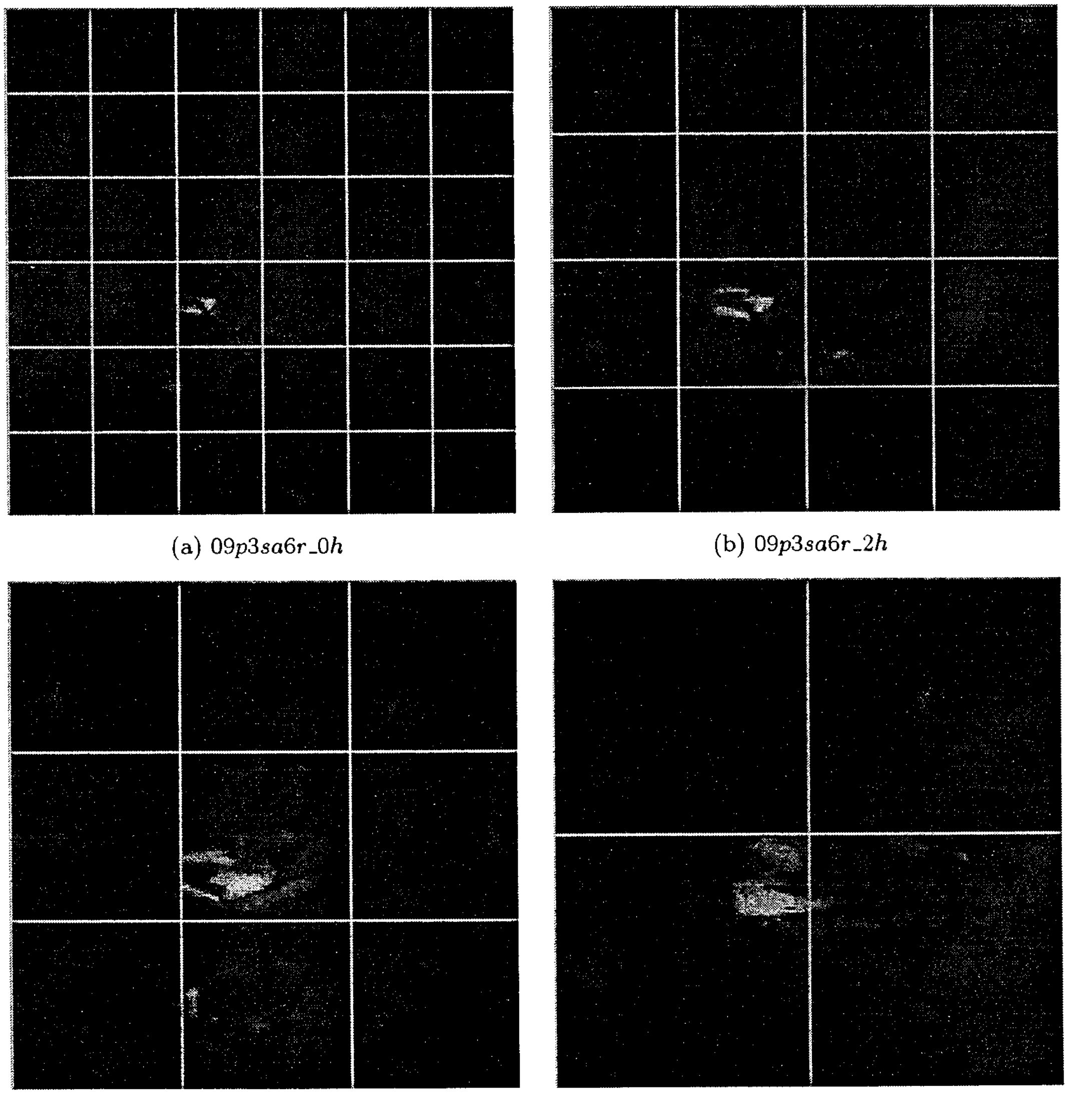
   where Nb is the total number of feature regions within the testing image.

7.5.5 Experimental Results

After the BMB is constructed through applying the supervised SOM algorithm with 20 training images, another twenty FLIR images were used as testing images for target detection experiment. Without knowing the validity scope of each BMB member, we first treated all five BMB members as equally important and assigned each one's weight to 0.2. For the total 217 feature cells in the 20 testing images, we achieved a 100% detection rate and a 12% false alarm.The corresponding confusion matrix is shown in Figure 7.13(a). Then, we use these 20 images to learn the validity scope for each BMB member using the SRV reinforcement learning algorithm. After 200 iterations, the detection false alarm decreased by 2%, and the new confusion matrix is shown in Figure 7.13(b).

7.5.6 Future Work

Our work on the supervised SOM algorithm and the SRV reinforcement learning algorithm has shown that, by introducing learning capabilities into an ATR system, we can build a statistical model for the complex natural background from real images and improve it as we feed the system with more examples. Our future work will focus on improving these two algorithms. First, we need to make the supervised SOM algorithm incremental, which would allow the system to process new examples more efficiently. Second, we need to make modifications to the SRV algorithm so that we could (1) abandon the assumption that each BMB member has a normal distributed validity scope (Equation 7.9, 7.10). (2) use more sophisticated functions to approximate the relationship between contextual vectors and the two internal state vectors Φ and Ψ, not just a linear function.



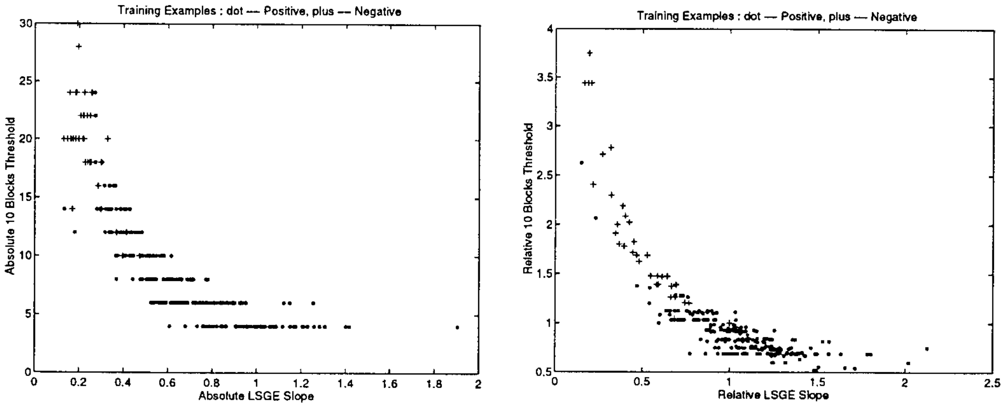
(a) 09p3sa6r_0h

(b) 09p3sa6r_2h

(c) 09p3sa6r_4h

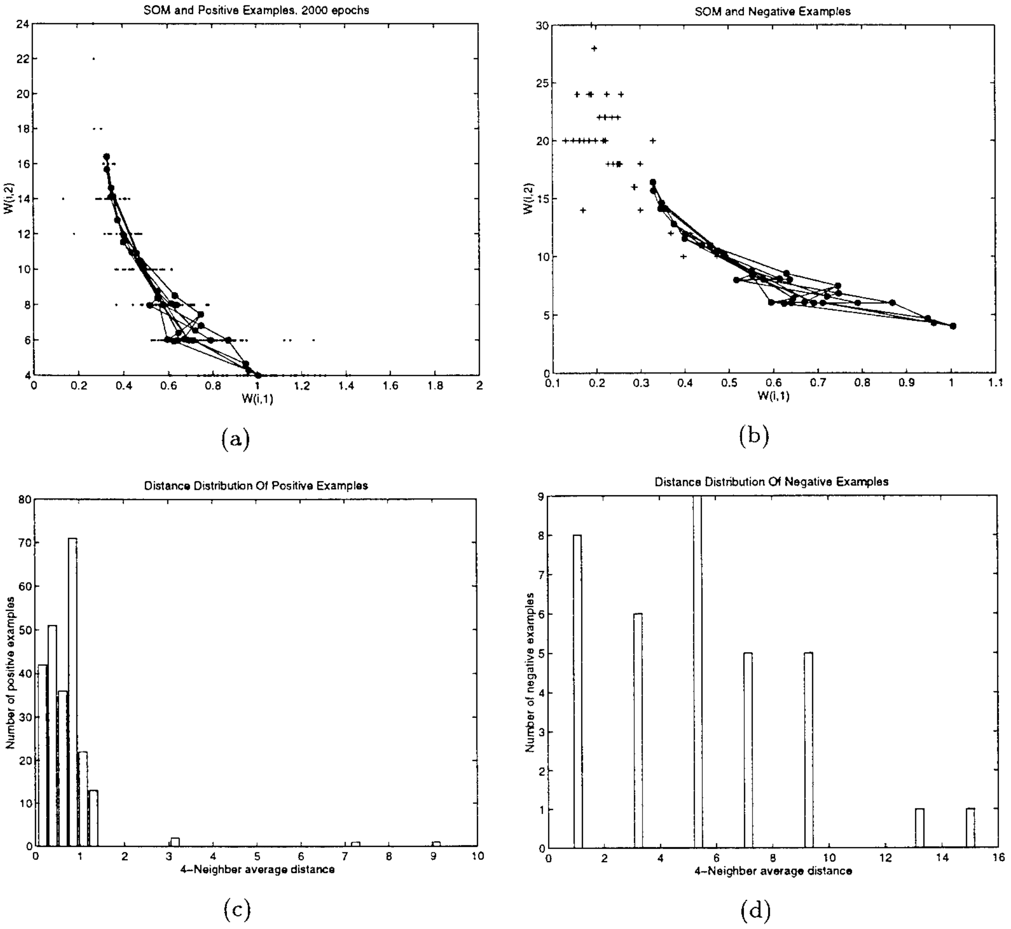
(d) 09p3sa6r_6h

Figure 7.7: Examples of training images.



(a) (b)

Figure 7.8: (a) Feature distribution for absolute LSGE (b) Feature distribution for relative LSGE.



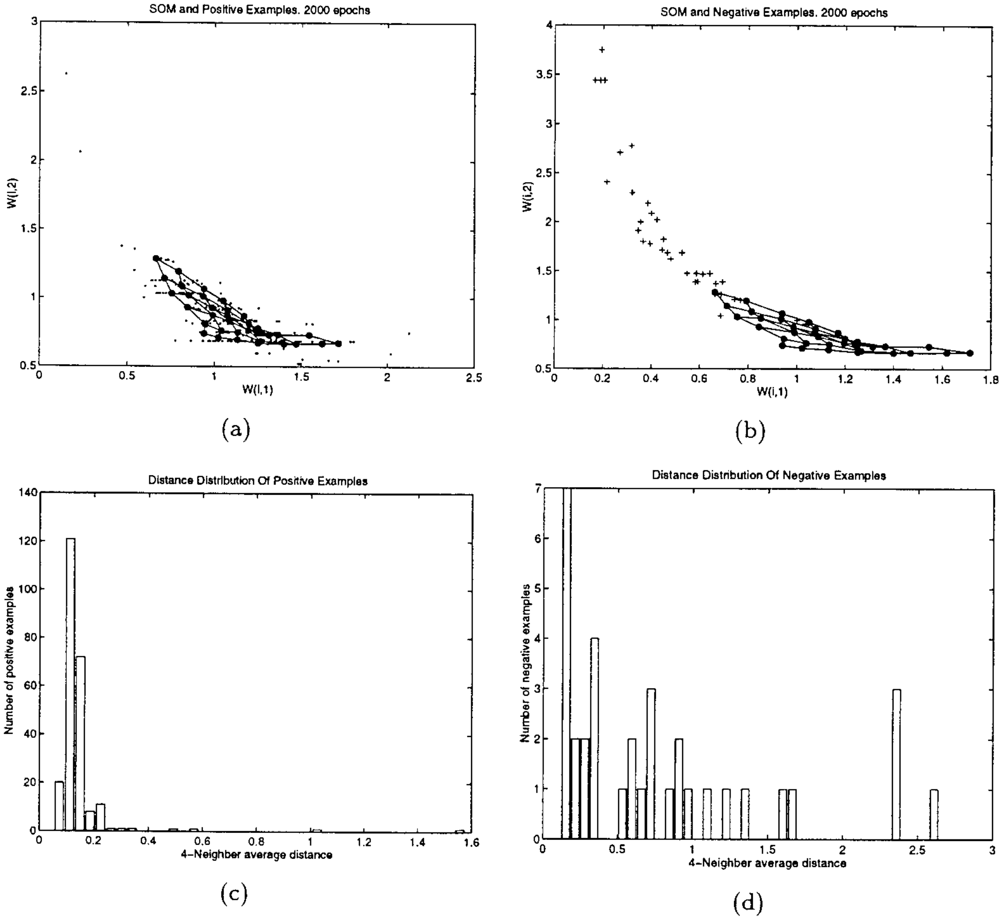
(a)

(b)

(c)

(d)

Figure 7.9: SOM constructed using kohonen's algorithm and absolute LSGE feature data. (a) SOM overlapped with positive examples (b) SOM overlapped with negative examples (c) distribution of $4 - Neighber$ distance of positive examples. (d) distribution of $4 - Neighber$ distance of negative examples.



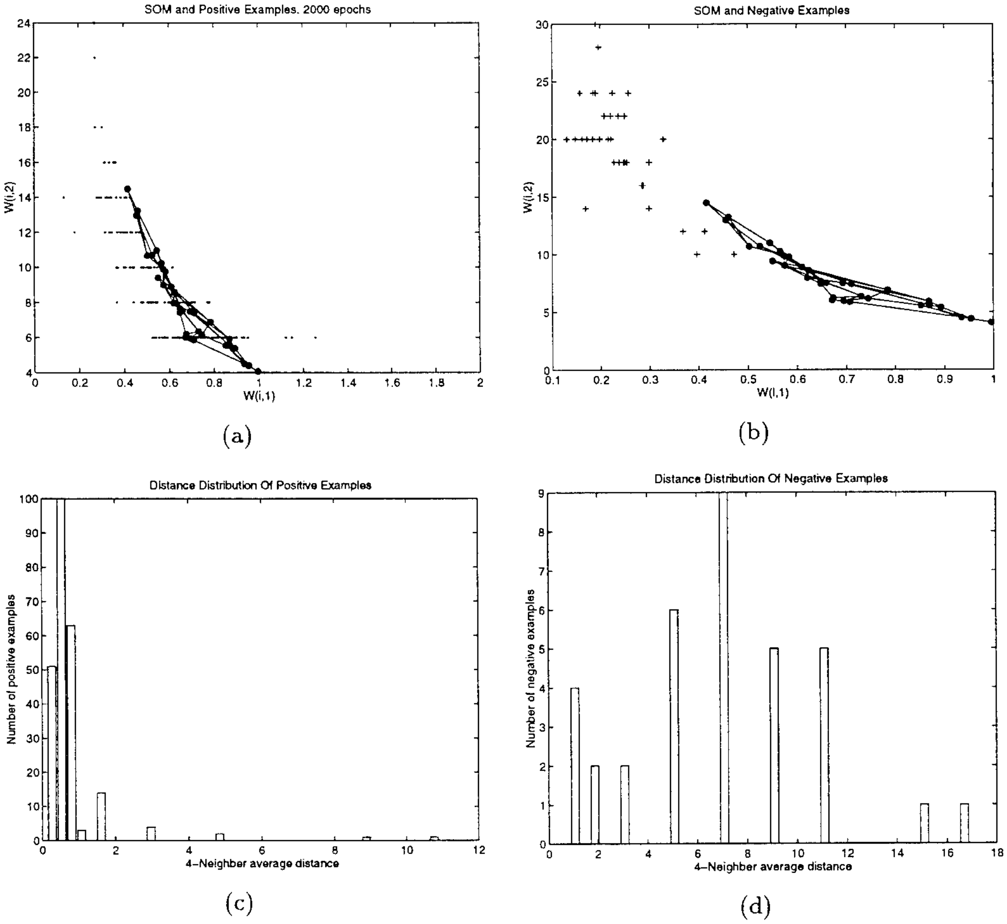
(a)

(b)

(c)

(d)

Figure 7.10: SOM constructed using kohonen's algorithm and relative LSGE feature data. (a) SOM overlapped with positive examples (b) SOM overlapped with negative examples (c) distribution of $4 - Neighber$ distance of positive examples. (d) distribution of $4 - Neighber$ distance of negative examples.


SOM and Positive Examples. 2000 epochs

SOM and Negative Examples

(a)

(b)

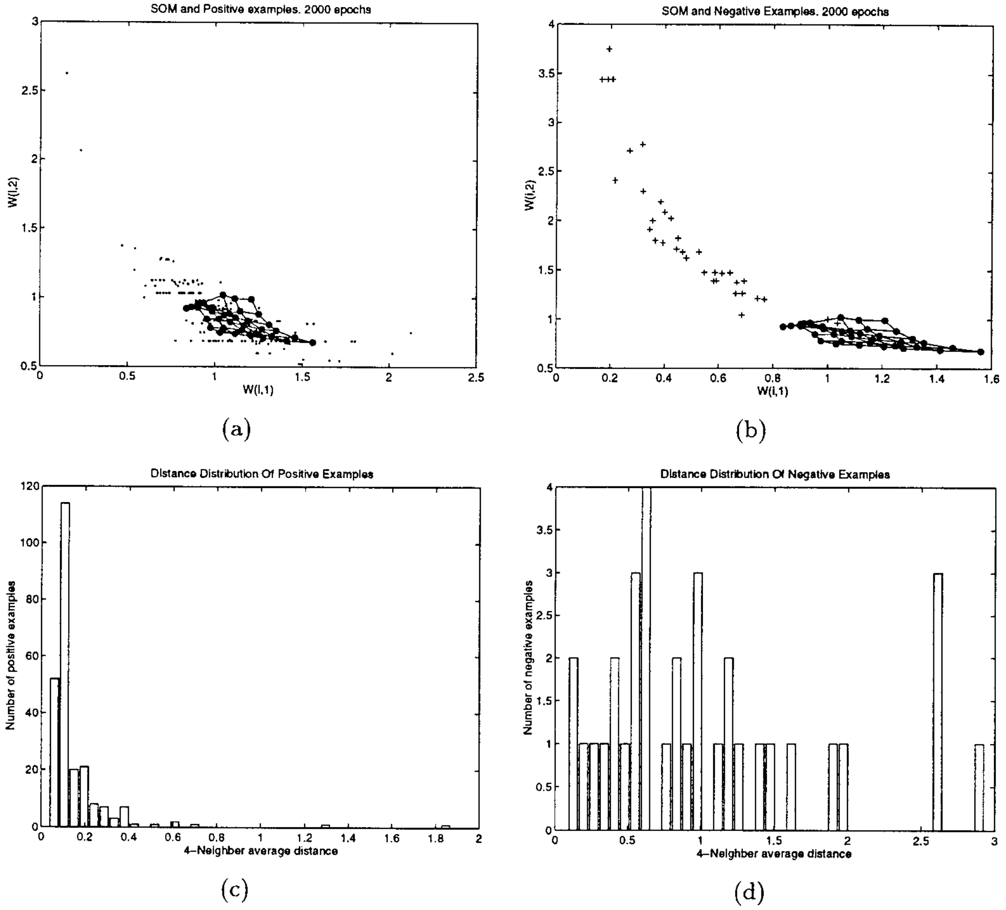
Distance Distribution Of Positive Examples

Distance Distribution Of Negative Examples

(c)

(d)

Figure 7.11: SOM constructed using supervised SOM algorithm and absolute LSGE feature data. (a) SOM overlapped with positive examples (b) SOM overlapped with negative examples (c) distribution of $4 - Neighber$ distance of positive examples. (d) distribution of $4 - Neighber$ distance of negative examples.



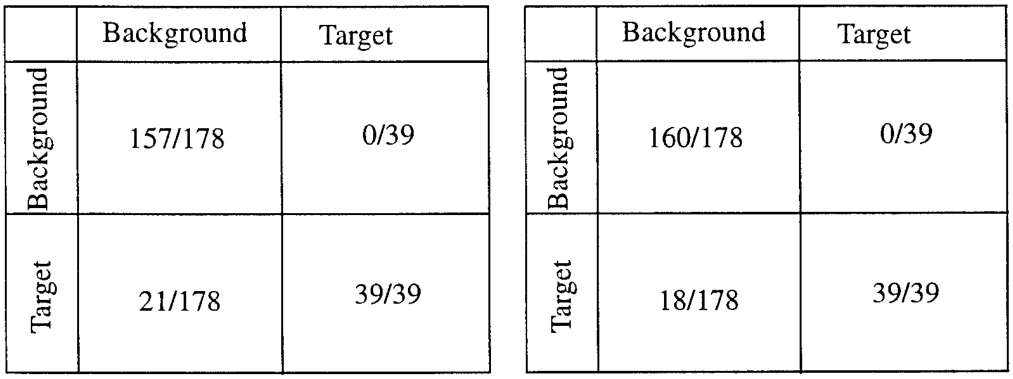
(a)

(b)

(c)

(d)

Figure 7.12: SOM constructed using supervised SOM algorithm and relative LSGE feature data. (a) SOM overlapped with positive examples (b) SOM overlapped with negative examples (c) distribution of $4 - Neighber$ distance of positive examples. (d) distribution of $4 - Neighber$ distance of negative examples.



	Background	Target
Background	157/178	0/39
Target	21/178	39/39

(a)

	Background	Target
Background	160/178	0/39
Target	18/178	39/39

(b)

Figure 7.13: Confusion matrix of the detection experiment. (a) before the reinforcement learning of the validity scopes. (b) after the reinforcement learning of the validity scopes.

Chapter 8

Case-Based Learning of Recognition Strategies

8.1 Introduction

Photointerpretation (PI) has been an important application domain of image understanding (IU) techniques for about two decades. An important goal of PI or image exploitation (extraction of intelligence from image data, particularly aerial imagery) is to aid reconnaissance tasks, such as airfield, port, and troop movement monitoring. The problem of PI is one of identifying instances of "known" object models in images acquired from a platform, such as by a satellite or a reconnaissance aircraft. Like PI, automatic target recognition (ATR) is also concerned with finding instances of known targets in the input sensor data. Model-based object recognition is a challenging task under real-world conditions such as occlusion, shadow, cloud cover, haze, seasonal variations, clutter, and various other forms of image degradation. Additionally, ATR scenarios are characterized by multi-modal imagery, low resolution, and camouflage. All of these problems put heavy requirements on any IU system to be robust.

Automatic acquisition of recognition strategies in dynamic situations has been a bottleneck in the development of automated IU systems applied to real-world problems, such as PI and ATR. The problem occurs while matching a stored object model to an input instance of that model and is attributed to the initially unknown pose of object and the varying environmental conditions. During the process of image/scene understanding, a human relies heavily on the memory of past cases and experience. We use the Case-Based Reasoning (CBR) paradigm in which "past" experiences are stored in memory as cases and are used to solve a new problem case. Similar cases can be combined to create problem solving shortcuts or to anticipate problems in new situations. The set of cases is prioritized and a strategy for the current problem is generated and executed. Various combinations of cases are created until a successful solution is reached.

8.2 Learning Recognition Strategies

Figure 8.1 describes our approach to learning recognition strategies for real-world object recognition tasks. The main learning paradigm employed in our recognition scheme is Case-Based Reasoning. The detailed CBR-based recognition framework shown in Figure 8.1 consists of four subtasks: (a) the generation of goal-directed recognition strategies using CBR, (b) the construction and maintenance of the Generalized Case Library (GCL) that collects past situations and corresponding actions, (c) the development of efficient algorithms for matching new situations to previous cases, and (d) the generalization of new cases using a variation of Explanation-Based Learning (EBL). Additionally, our approach also addresses the problems of indexing into the object model data base and the verification of object hypotheses. This latter task consists of two main parts: (a) the creation and refinement of the decision structures for indexing, using a variant of the Conceptual Clustering (CC) learning technique, and (b) the implementation of the indexing and matching algorithms. In this report, we focus on the CBR-based framework.

8.2.1 Case-based reasoning (CBR)

Case-based approaches are characterized by how the learner represents what it has learned so far, as well as the analogical methods which are used to transfer the learned experience. Human expertise in problem solving is largely dependent on past experiences. This idea has influenced the evolution of Case-Based Reasoning [7, 57, 88]. A related approach is that of reasoning by analogy [6, 40]. In CBR, "past" experiences are stored in memory as cases and are used to solve a new problem case. Given a problem to be solved, the case-based method *retrieves* from the memory the solution to a similar problem encountered in the past, *adapts* the previous solution to the current problem, and *stores* the new problem-solution packet as another case in the memory.

There are several advantages of CBR as a learning paradigm. First, CBR has the capability of anticipating and therefore avoiding past mistakes as well as focusing on the most

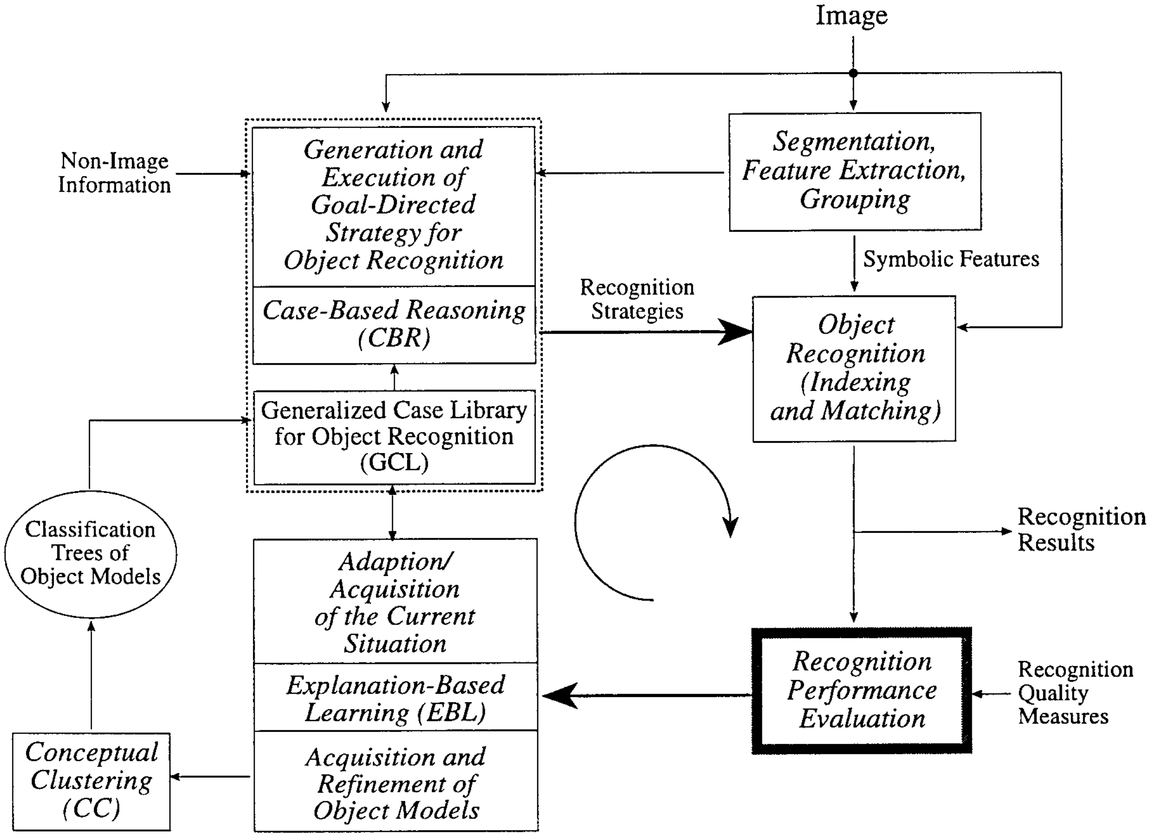


Image
Non-Image Information
Generation and Execution of Goal-Directed Strategy for Object Recognition
Case-Based Reasoning (CBR)
Generalized Case Library for Object Recognition (GCL)
Segmentation, Feature Extraction, Grouping
Symbolic Features
Recognition Strategies
Object Recognition (Indexing and Matching)
Recognition Results
Classification Trees of Object Models
Adaption/ Acquisition of the Current Situation
Explanation-Based Learning (EBL)
Acquisition and Refinement of Object Models
Conceptual Clustering (CC)
Recognition Performance Evaluation
Recognition Quality Measures

Figure 8.1: A CBR framework for learning recognition strategies. EBL generalizes cases and along with CC it facilitates automatic knowledge acquisition of object models.

important aspects of a problem first. All of these lead to an increase in efficiency over time. Second, the learning process is fairly uncomplicated, since CBR does not require causal models like inductive learning or extensive domain knowledge like analytic learning. Third, the individual or generalized cases can also serve as explanations. Fourth, the process is scalable. Fifth, the knowledge acquisition bottleneck is relatively simple to solve in CBR than in conventional learning systems. This is because individual cases interact a little among themselves unlike the rules. The major concerns with CBR are the selection of the indexing scheme to organize cases in the memory, the method for choosing the most relevant cases at reasoning time, and the adaptation heuristics to modify previous cases to fit the current problem.

There are two major types of case-based approaches: *interpretive/classification* (or precedent-based) CBR, and *problem solving* CBR. In the precedent-based CBR, the task is to decide whether or not a new case should be treated like one of the stored cases based on similarities and differences between the two. This is done by generating a pro's and con's analysis from a comparison of the two cases. In problem solving CBR, a solution for the new problem is formulated by suitably modifying past solutions. In either approach, a proposed solution must be verified for appropriateness. This is particularly important if the derived solution is based on "unexplained" experiences. This verification process is akin to an evaluation procedure associated with any learning process. An interpretive CBR is used in such evaluation process to provide a check on the use of knowledge derived from experience.

8.2.2 CBR in IU

Current model-based IU approaches to object recognition generally utilize only the geometric descriptions of object models, i.e., they emphasize the recognition problem as a characteristic of individual object models only. However, there are various factors, such as contextual information, sensor type, target type, scene models, and related non-image information that may influence the outcome of recognition in real-world applications such as ATR, PI, navigation. Humans also rely on such ancillary information for object recognition and scene understanding. For example, it is well known in the intelligence community that oxen yoked to water pumps in Southeast Asia resemble anti-aircraft artillery in aerial images [2]. Thus, without the knowledge of the area being examined, an image analyst or an automated PI system may be misled easily. Thus, prior experience in addition to object/sensor models is important for devising efficient and robust recognition strategies to deal with noisy data or occluded targets against complex backgrounds.

Prototypical situations (cases) observed in the past are useful for the recognition of objects as well as for the assessment of entire scenes. An example of a case in the PI context is given in Figure 8.2. Each path from the root node to a leaf node in the tree represents a single case. The path incorporates the information normally used at each level in an object recognition task, e.g., aircraft recognition. It includes contextual information, e.g., airfield, scene type, e.g., tarmac parking areas, the best object recognition strategy, e.g., selection of segmentation, feature extraction, recognition algorithms and their parameters, and corresponding image analysis goals, e.g., finding instances of transport aircraft such as *Hercules*. A case of ATR would additionally include sensor type, terrain, and radiometric information.

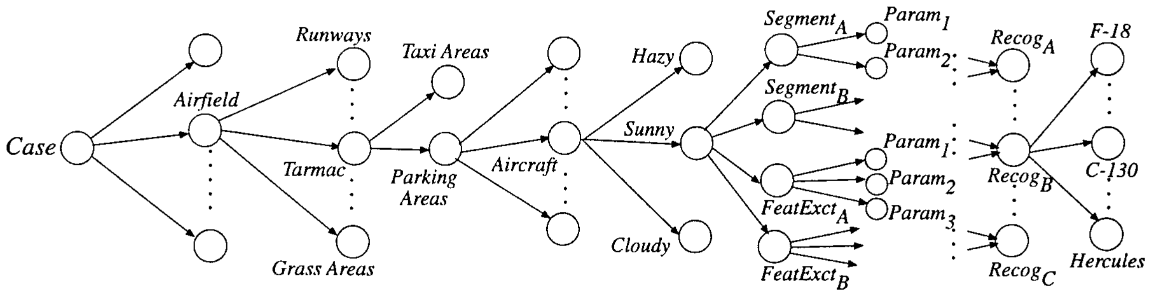


Figure 8.2: Representation of a case in the photointerpretation context.

Case-based methods are best suited to problems for which *many* training cases are available, perhaps with many *exceptional* cases, and it is *difficult* to specify appropriate behavior using abstract rules. Most IU applications, such as ATR and PI, are characterized by large-volume image exploitation corresponding to a variety of scenarios, many of which require unique analysis. Besides, IU for unstructured environments is difficult to formalize in terms of rules that are general enough to be applicable to diverse situations. For example, recognition of a *Hercules* aircraft in a parked area of the tarmac under sunny condition has been successful in the past by following the path from the root node to the leaf marked "hercules" in the case representation of Figure 8.2. However, the same path may not lead to a successful recognition of an F-18 aircraft. Thus, the *case* of recognizing a *Hercules* is not the same as that of an F-18.

8.2.3 Learning method

The learning approach is concerned with (a) building new cases, (b) generalizing and refining existing cases, for a particular application. As indicated in Figure 8.1, the relevant knowledge is accumulated in the generalized case library. For updating and indexing into the GCL we use a combination of two different learning strategies: CBR is used primarily for retrieving the relevant earlier experiences and updating (restructuring) the knowledge base; CC is used for maintaining decision structures (classification trees) that allow efficient object recognition at run time.

The GCL is the collection of knowledge that allows the system to perform object recognition and scene assessment. It is a dynamic body of information that represents the experience base of the object recognition system. For efficient indexing, the GCL is represented as a structured hierarchy of individual cases. Each case, in turn, is represented using scripts and memory organization packets (MOPs) which are meta-scripts [97, 96]. These data structures are appropriate for episodic memory or time sequences of episodes which are equivalent to the sequences of computational steps/recognition strategies in our case. Since scripts contain more specialized information, these are used for lower-levels of a case structure. The MOPs allow representation of more generic knowledge such as an airfield which can be instantiated and specified for recognition of multiple aircraft types.

When a new problem situation or IU task is encountered, e.g., recognition of aircraft on tarmacs, the process of interpreting and assimilating the new task in CBR framework breaks down into the following steps:

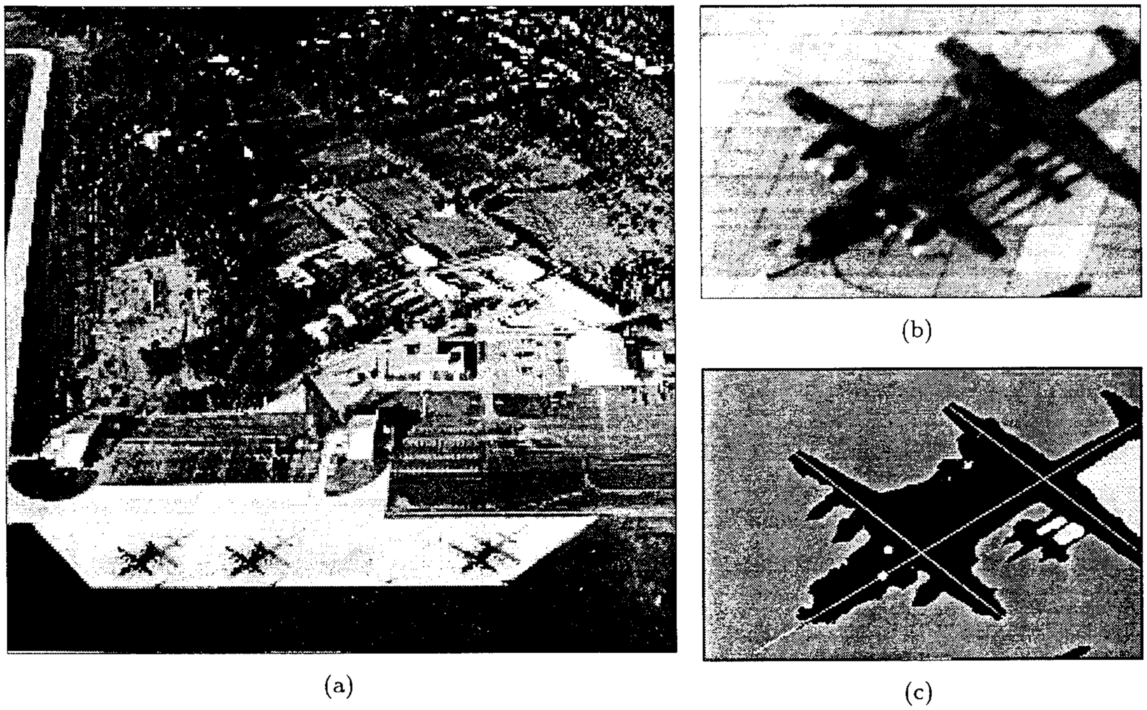
- *Assign Indices* – Features of the new task are assigned as indices characterizing the task. For example, "tarmac" and "aircraft" can be used to characterize the task as "aircraft-on-tarmac" which will be a particular subtask of "aircraft-in-airfield" task.
- *Retrieve* – The indices are used to retrieve from memory a similar case encountered in the past based on similarities and differences. The past case contains the prior solution. For example, a case which has involved aircraft on tarmac instead of grass areas.
- *Modify* – The previous solution is adapted to the current task, resulting in a proposed solution. For example, the previous recognition may have occurred under sunny conditions which required detection of shadows, while the weather condition for the current task is cloudy. Thus, the previous case is modified by eliminating all computational steps involving shadows.
- *Test* – The proposed solution is carried out. It may lead to success or failure. For example, the parameters of the segmentation algorithm for detecting regions of interest may have been retained as the same as in the previous case. On the other hand, the contrast of the current image may be low due to cloudy weather condition, thereby, requiring somewhat different segmentation parameter set.
- *Assign and Store* – If the solution succeeds, then indices are assigned to it and the solution is stored as a working solution. The successful plan is then incorporated into the case memory. If the solution is not too different from the proposed solution, then it affects the script of the existing case a little.
- *Explain, Repair, and Test* – If the solution fails, then the failure is explained, the working solution is repaired, and the test is again carried out. The explanation process identifies the source of the problem. For example, new segmentation parameters

> are selected when recognizing aircraft under cloudy weather condition. The predictive features of the problem are incorporated into the indexing rules to anticipate this problem in the future. For example, "aircraft-on-tarmac" index is extended to "aircraft-on-tarmac-sunny" and "aircraft-on-tarmac-cloudy." The failed plan is repaired to fix the problem, and the revised solution is then tested. The rest of the plan is carried out with new segmentation parameters in our example. A new case is then created in the memory to handle this new situation.

The results of the CBR-generated strategy are passed to the interpretation and evaluation component. Case indexing and matching is performed using the intermediate visual concepts. The different recognition states are: complete recognition, incomplete recognition, object occlusion, object model acquisition, object model refinement, and recognition failure. Now, three situations may arise. First, if the strategy is very similar to one of the cases extracted from the GCL, no learning takes place. In this instance, the system has encountered an "ordinary" image interpretation task in which the current collection of system knowledge is adequate. Second, if the strategy is an extension of an existing case (i.e., the existing case represents a subset of elements of the new strategy), a case refinement operation may be necessary. The new strategy and its associated case are sent to the EBL module to determine if any new information should be included in the existing case. Third, if a unique combination of existing cases has been utilized to create a novel strategy for a given problem, a case acquisition operation is required. The new strategy is passed to the EBL, which applies its system control knowledge in order to remove irrelevant details and conceptualize the scope of the strategy. This new strategy is then inserted as a new case into the GCL. The CBR and the EBL paradigms are combined in a complementary manner. CBR has the ability to index into a large number of potential solutions and select a set of cases that match the characteristics of the current object recognition task. However, the performance of CBR degrades with the size of the case library and also by the amount of irrelevant detail retained in the stored cases. EBL compensates for this by learning only the concepts underlying the individual cases before adding the conceptual abstraction of the cases to the GCL. On the other hand, since CBR combines a set of previous cases to create a single new case for the current problem, any bias of the EBL component towards a particular training example will be greatly reduced. In summary, CBR allows the capture of context and domain-specific information to improve recognition performance over time.

8.2.4 An Example

An example that illustrates the use of CBR for high-level object recognition is given in Figures 8.3–8.4. A knowledge-based technique initially identifies several regions of interest


(b)

(a)

(c)

Figure 8.3: High-level object recognition based on CBR. (a) Original image; (b) Initial region of interest (ROI); (c) Extracted dominant axes.

(ROIs) in the image that are likely to contain aircraft. One such ROI and its corresponding segmentation results are shown in Figures 8.3(b) and 8.3(c), respectively. Also shown in Figure 8.3(c) are the dominant axes of an aircraft structure along the wing and the fuselage. (The third axis corresponding to the shadow of the wing is found to be part of a shadow region and is removed subsequently.) The most "salient" features (with regard to edge strength and global connectivity) and the identified shadow lines are shown in Figures 8.4(a) and 8.4(b), respectively. Notice that most of the front edges on both wings are missing from the extracted line group.

A composite structure detection step identifies trapezoid-like shapes that are characteristic of wings, tails, and rudder in non-shadow lines (Figure 8.4(b)). Next, an evidence-based dynamic reasoning process seeks to instantiate one of these composite structures (that are aligned with the dominant axes) as a wing. This situation is shown in Figure 8.4(c). The

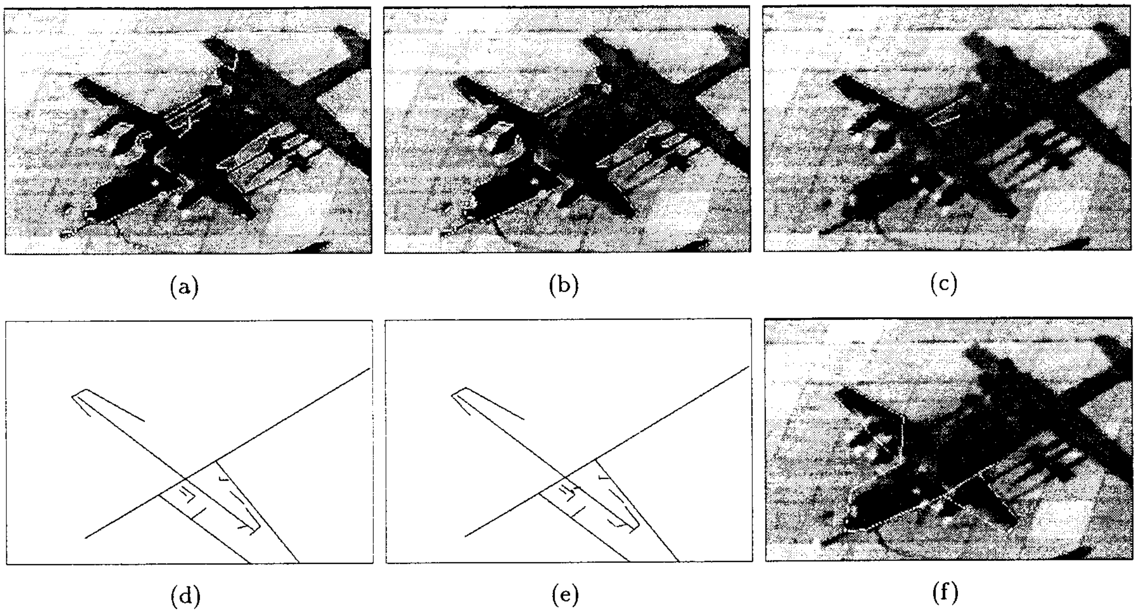


Figure 8.4: High-level object recognition based on CBR *(continued)*. (a) Fitted straight lines; (b) Detected shadow lines; (c) Trapezoid shapes in non-shadow groups; (d) Hypothesized right wing and projected left wing; (e) Emergence of additional non-shadow lines; (f) Final recognition result.

support for this hypothesis, however, is weak, as there is no evidence for the other wing (i.e., no trapezoid-like structure was detected that is aligned with the same dominant axis). Subsequently, less "salient" line features are acquired (Figure 8.4(e)) and a trapezoid-like structure is detected by relaxing the thresholds of the perceptual grouping process. The final recognition result is shown in Figure 8.4(f).

The experiences gained in this recognition "case" are:

- Shadow and object regions are similar (Figures 8.3(a)–(a)), therefore the rear part of the aircraft could not be recovered (Figure 8.4(f)) without using sensor/platform information.

- Relative positions of the sun and the sensor had given rise to specularity along the leading edges of the wings, making these hard to detect from edge information (Figures 8.4(a) and 8.4(d)).

- Evidence of engines had been helpful in hypothesizing a wing (Figure 8.4(d)).

Additional information in this case includes the sun angle, sensor position, sensor/platform parameters, segmentation parameters, directions of shadow regions in a ROI, etc. Clearly, such a "case" is valuable when the task is to investigate another ROI, say the one next to the current one in Figure 8.3(a) which contains another aircraft of the same type (i.e., a *Hercules*). The recognition algorithm will use the same segmentation parameters, will try to verify the front parts of the airplane first, and will know that the leading edges of the wings may be difficult to detect.

8.2.5 Implementation Issues and Performance Evaluation

There are several issues of practical importance in implementing a CBR-based recognition system. These issues are,

- representation and contents of a case in the memory,
- memory organization and selection of indexing rules and search algorithms,
- incorporation of changes over time in the cases and the indexing rules,
- recognition of a new situation as similar to a previous case, i.e., the choice of similarity metrics,
- adaptation of old solutions to new problems, i.e., selection of modification rules,
- acceptance or rejection of a new case that is in conflict with a previous case, i.e., explaining the differences between two problem situations,
- learning from mistakes and devising the repairing rules.

Unlike the rule-based systems, the rules for indexing, modification, and repair do not make up the principal knowledge base but, rather, independent support modules. Thus, the complexity involved is less severe than in most rule-based systems. However, the theory of case-based reasoning suggests that these rules would themselves be acquired by experience from cases through a recursive application of the CBR algorithm. That is, the system would derive rules for indexing, modification, and repair from cases and experience.

The evaluation of the performance of a CBR system can be quite complex due to the nature of the represented knowledge. One way to express the recognition success would be to note the similarity between two problem situations. If these situations are identical, then one would expect identical recognition results. The performance difference would increase with the difference in the situations. Finding a single difference (or similarity) metric would be quite complex as there may exist a number of alternatives to compare two situations. Thus, a multi-objective criterion function would be more appropriate. One could simply focus on the various rules for indexing, modification, and repair to evaluate the performance of a CBR system. For example, the hit vs. miss ratio in retrieving cases from the memory using the indexing rules can be one measure. Various tools from the field of memory management can be used as potential measures to evaluate the efficiency of memory management in a CBR system, e.g., memory usage, memory fragmentation, distributed vs. centralized memory, dynamic memory organization.

8.3 Future Work

Our initial goal of learning recognition strategies using case-based approaches would be limited to PI applications. We have already developed an aircraft recognition system for this purpose and are in the process of extending it further. Currently, this system can handle quite complex imagery and the variabilities present in such images would be ideal for a case-based approach. We have presented some results using this system in this report and sketched our case-based approach. Since our focus is on developing recognition strategies through a learning process, we are minimizing our effort to design appropriate CBR tools. We have experimented with a LISP-based system for CBR. Our future effort is directed towards developing (a) a prototype system which will have all the basic elements of CBR and (b) reasoning, adaptation and indexing approaches that will make CBR an effective approach for IU applications.

Chapter 9

Learning Composite Visual Concepts

9.1 Introduction

The context of the learning problem addressed here is *structural object recognition*, which is based on the assumption that structural primitives, extracted from the image in a bottom-up fashion, can be used to describe and recognize the objects of interest. The main advantage of this approach is that it facilitates (at least in principle) recognition under object and aspect variations and, as a recognition-by-components approach, under partial occlusion.

The main problems associated with the structural recognition approach are (a) the computational expense for matching structural object descriptions, (b) the reliable extraction of structural primitives from the image, and (c) the descriptive limitations of the commonly used structural features. The combinatorial problems associated with matching structural descriptions call for methods to limit the search space. When object models are complex, their direct instantiation, either in a top down or a bottom-up, becomes impractical. A logical solution is to describe objects as assemblies of smaller substructures (intermediate visual concepts) that can be instantiated with much less effort. Perceptual grouping methods (e.g., [63, 86, 94]) make use of this fact by using simple geometrical relationships (e.g., collinearity, cotermination, parallelism, etc.) to assemble primitives into more complex features. However, due to the domain-independent specification of perceptual groupings, their "indexing power" is insufficient in applications with more than a few object categories. Another weakness of current structural recognition techniques is their reliance upon a single type of primitive feature, which leads to low redundancy and inappropriate descriptions.

We address the first problem by *learning* significant composite structures that are hierarchically assembled from geometric primitives and serve the purpose of intermediate goals for partial recognition. The other two problems are approached by using a larger variety of *different* structural feature types and corresponding object representations, thus achieving a higher level of redundancy. For the recognition framework we adopt a model-based hypothesize-and-test approach that consists of three main steps: primitive extraction, model-base indexing, and model verification. These three steps operate in a *bootstrap fashion*, i.e., the process starts in a bottom-up mode by extracting primitives and combining them in a meaningful way up to a point when a plausible object hypothesis can be made. Then the recognition process turns into a goal- (model-) directed search and verification process.

The bottom-up part of the recognition process can be viewed as a multi-stage grouping process. At the lowest level, individual pixels are grouped to form the structural primitives, e.g., straight line segments, arcs, regions, etc. At the intermediate-level, the structural primitives produced by feature extraction are combined into more complex structural arrangements, usually biased by perceptual (i.e., domain-independent) constraints. The main goals of the second grouping step[1] are to

1. combine structural features in a way that they are likely to belong to the *same* object, thus reducing the number of "clutter" features that have no correspondence in the model structure and

2. to produce more expressive, object-specific entities that allow effective indexing into the model base.

It is the second item that is our main focus in this part of the project. We need to ask the question, which properties, apart from being perceptually significant, should be incorporated into the grouping process. We believe that, in order to lead to useful object indices, this second set of grouping criteria cannot be model- or domain-independent but needs to be adjusted to the particular application domain, the objects involved, and the context in which they appear. The value of a particular feature group depends mainly upon (a) its *indexing power*, i.e., its capability to select a specific object (or a small set of objects) and (b) its *operationality*, i.e., the effort needed to instantiate it. The general approach for the use of learning to come up with the most effective feature groupings is described in the following.

[1] This step is the one commonly referred to as "grouping."

9.2 General Idea

The intermediate-level part of the project is focused on the problem of "inventing" new composite structural features (intermediate visual concepts) to improve recognition performance. We use intermediate visual concepts that are directly related to the application domain. For this purpose, we select certain high-order assemblies of primitive features which are both perceptually salient and sufficiently distinct to allow very efficient indexing. We employ a two-step grouping strategy that consists of

1. a domain-independent perceptual grouping stage (which ensures perceptual saliency of the selected groups to cope with over-segmentation), followed by

2. a model-based grouping process that is domain-dependent. The high-order, model-based groups are formed as assemblies from the lower-order perceptual groups.

Current perceptual grouping methods (e.g., [63, 86, 94]) are based on (a) a single type of primitives and (b) grouping rules that are predetermined and not adapted to the application domain. The use of a single feature type has the advantage of simple representations and grouping criteria that can be evaluated efficiently. Also, the corresponding structural descriptions are independent of the problem domain. The disadvantages are that

1. the perceptual "saliency" of groupings between different types of primitive features is not used,

2. groupings based on a single feature type are inherently brittle, and

3. fixed, domain-independent grouping rules are not suitable for dynamically changing scenes.

In our approach, we combine multiple types of structural features at the intermediate level, such as line segments, conic sections, corners, inflection points, blobs, etc., in order to increase the descriptive power and robustness (through higher redundancy) of the "polymorphic" feature groupings. The problem of grouping polymorphic features is more challenging than grouping features of the same kind, with regard to the representations and grouping algorithms involved.

The selection and generalization of the intermediate visual concepts is critical in order to in-sure optimal recognition performance. It requires knowledge of the application domain, the imaging process, the behavior of the perceptual grouping stage, and the recognition utility of the intermediate visual concepts. We use Explanation-Based Learning (EBL) to solve this special knowledge acquisition problem. EBL is useful in this context to detect inherent pattern regularities and to generalize patterns, i.e., to determine the simplest description with respect to a given set of operators. In summary, the strategy at this level involves:

1. The use of a two-stage grouping strategy that involves (a) perceptual grouping and (b) model-based grouping with a database of generalized visual concepts.
2. The use of EBL to automatically infer the most useful intermediate visual concepts by applying the entire recognition "engine" to real examples.
3. The use of "polymorphic" feature groupings based on multiple feature types.

The main advantages we expect from this strategy are:

1. A significant reduction of the overall search complexity for structural model instantiation by using high-order intermediate visual concepts.
2. Increased robustness and indexing power from the use of polymorphic groupings.
3. Adaptation of grouping processes to application domains and environmental conditions.

9.2.1 Example

In the aircraft picture shown in Figure 9.1 it is evident that the groups of lines that compose the wings, tails, and rudders, form high-order groupings that are characteristic for many types of aircraft. Obtaining a conceptual description of certain configurations, e.g., the trapezoid that forms the wings, is useful for improving the recognition of other aircraft.

9.2.2 Goals

The main goals at the intermediate level are to automatically acquire new visual concepts from examples, using Explanation-Based Learning and incorporating polymorphic feature groupings. We shall demonstrate that the use of domain- and object-specific grouping, in combination with traditional perceptual grouping, can significantly improve the efficiency of indexing and object recognition.

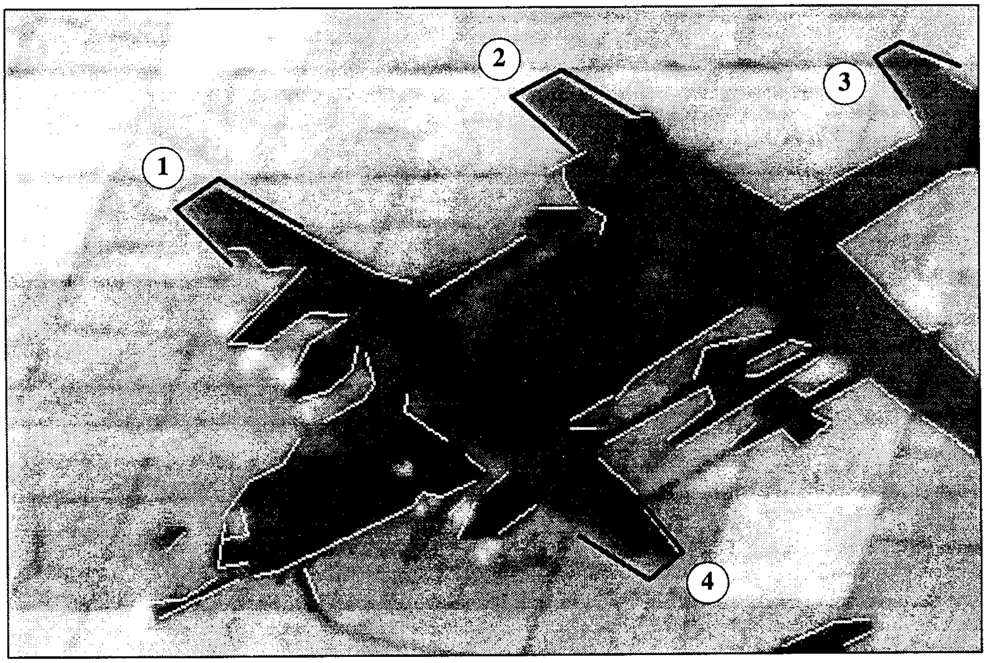


Figure 9.1: Domain-specific, composite visual concepts are formed by combining perceptually salient low-order groupings. Here only straight line segments are used as initial primitives. An example for a simple intermediate-level concept is the typical trapezoid shape found at the ends of the aircraft wings. Four instances (1–4) of this concept are outlined and marked in this image.

9.3 Approach

The instantiation of visual concepts is performed in a two-stage process (Figure 9.2). Initially, the simple features extracted from the input image by various different selection mechanisms (e.g., straight line segments, conic segments, homogeneous blobs, etc.) are grouped using domain-independent perceptual grouping criteria. Examples for the grouping criteria are collinearity, cotermination, parallelism, proximity, relative size, symmetry.

At the second stage, domain-specific models of high-order composite structures (intermediate visual concepts) that have been found useful for recognizing objects guide the grouping process. Visual concepts are learned by the system (see below) and stored in a local database that is continually updated. Only those groupings are considered here that were found perceptually significant at the initial perceptual grouping stage. During actual

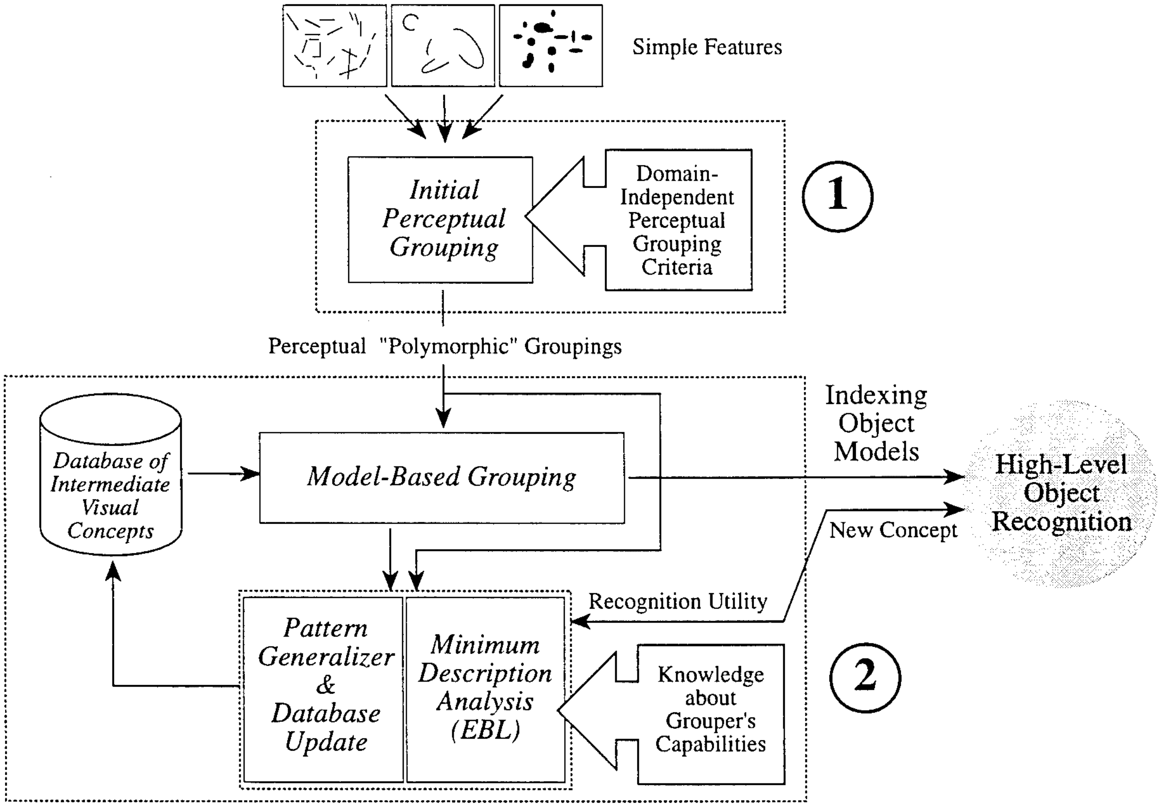

Simple Features
Initial Perceptual Grouping
Domain-Independent Perceptual Grouping Criteria
1
Perceptual "Polymorphic" Groupings
Database of Intermediate Visual Concepts
Model-Based Grouping
Indexing Object Models
High-Level Object Recognition
New Concept
Recognition Utility
Pattern Generalizer & Database Update
Minimum Description Analysis (EBL)
Knowledge about Grouper's Capabilities
2

Figure 9.2: Learning intermediate visual concepts using Explanation-Based Learning (EBL)

(routine) recognition, the visual concepts found at this stage are directly used for indexing into the object model base.

Learning of new visual concepts is based on the following criteria:

Perceptual saliency: A concept must be perceptually salient, i.e., receive a high score in the first (perceptual) grouping stage.

Operationality: A concept must be describable in terms of the operators that the model-based grouper is able to perform. For this purpose, knowledge about these operators is supplied in explicit form.

Simplicity: Concepts that permit a simple description (i.e., one with few grouping steps

/ transformations) are preferred. EBL is used to find the simplest description for a given feature configuration (*Minimum Description Analysis*).

Recognition utility: Only those concepts that are found to be useful in recognizing a particular object are eventually accepted. This is determined by considering the outcomes of the high-level recognition steps.

Visual concepts in the database are generalizations of the actually observed feature configurations, produced by analytic (EBL) learning (Pattern Generalizer). The representation of a concept in the database is an annotated symbolic description, which is generalized by parameterizing specific geometrical properties of the corresponding feature representation. The task of the Model-Based Grouper module is to instantiate the visual concepts, in the stream of perceptual groups, operating in a goal-directed fashion. The concepts (goals) are supplied to the grouper as decision structures that are updated dynamically when the contents of the database are changed. Interaction with high-level object recognition occurs in two forms. First, instantiated known groups can be directly used for indexing into the model base at the high level. (The association between intermediate concepts and object models is done at the high level.) Secondly, high-level recognition is invoked to determine the recognition utility of new concepts.

The use of a small set of fixed bottom-up composite structural concepts allows efficient detection in images. Similar arguments hold for top-down search for specific arrangements when the number of possible objects is small. The disadvantage of this approach is that a small but fixed set of intermediate structural concepts is generally not useful in different application domains. For using top-down, model-based composite structures, the number of models is restricted. In both cases, the manual specification of suitable intermediate structures is difficult.

The following specific tasks are involved:

9.3.1 Task 1 — Model-Based Interpretation of Perceptual Groups

We develop methods for collecting structural primitives of different types (e.g., lines, arcs, parametric curves, blobs) into polymorphic groups, using a set of perceptually significant spatial relationships. The relationships (e.g., proximity, collinearity, symmetry, relative size) being used depend upon the type of elements contained in each particular group. The purpose of this initial bottom-up grouping process is to supply an ordered set of composite structures that have a high probability of being semantically meaningful. The database of perceptual relationships used in this task is fixed, i.e., not subject to adaptation during runtime. However, this database must be designed to allow easy extension when new structural feature types are introduced. The main subtasks are to develop (a) the database of perceptual relationships, (b) evaluation function to measure the "saliency" of high-order polymorphic groups, and (c) efficient grouping algorithms that can handle polymorphic structures.

9.3.2 Task 2 — Composite Structure Model Acquisition and Refinement

We consider the actual semantic significance of perceptual groups with regard to the given application domain, in contrast to the previous task, where we employ only general perceptual cues. The module developed in this task uses the initial perceptual groups developed in Task B.1 for ultimately creating an index into the object model database. For this purpose, the module tries to form more complex groups from the incoming simple groups by using a database of semantically relevant structures. The database is created and maintained by a learning scheme based on Explanation-Based Learning (EBL). The major steps in this task are (a) the development of a suitable representation for high-order polymorphic feature groups which can also express their variability, (b) the adaptation of EBL for learning parameterized geometric concepts and its implementation in software, and (c) the development of efficient matching algorithms that can make use of the polymorphic nature of the feature groups.

9.3.3 Task 3 — Composite Structure Learning Subsystem

The goal of this task is the integration of all components needed for the adaptive intermediate-level learning scheme. Here we address in particular the interaction between the database of composite feature structures (Task 2) and the object models at the high level. The interaction with the high-level recognition module is needed to determine the utility of an observed feature structure for recognizing a particular object.

9.4 Learning at the Intermediate-Level Vision: Previous Work

Learning at the intermediate level has been applied mainly in the areas of texture recognition, algorithm parameter adjustment, motion perception, and specific vision tasks, such as road following. Currently, clustering methods are the most popular adaptation or learning paradigm at this level, followed by the use of neural networks and some applications of genetic algorithms. Structural learning methods, such as EBL or CBR are currently much less used at the intermediate level.

An example for inductive learning at the intermediate level is the approach to texture recognition described by Pachowicz [75]. He uses a scaling process to convert feature vectors of texture statistics into symbolic intervals and then applies an inductive learning program to find the most preferred symbolic expression according to a specified criterion. The method also employs a rule optimization technique after texture learning and prior to recognition to allow rule generalization. A performance improvement over the traditional nearest-neighbor clustering method is demonstrated.

Gillies [41] reports a learning system based on Genetic Algorithms for generating image domain feature detectors to find the location of objects in the image. A genetic search method is used to generate populations of feature detectors which are morphological operators. The functions performed by the layered system are tailored to the specific imagery on which the system is trained. The system is also shown to handle multi-class discrimination.

Another application of Genetic Algorithms at the intermediate level is the work done by Roth and Levine [90], which is a learning-based approach to extraction of geometric primitives (parametric curves) from images. In this approach, a geometric primitive is genetically represented by the minimal set of points instead of its parameters. Learning involves determining the minimal set of points for a given primitive type that optimally fits the data. Montana [69] reports an expert system for the interpretation of passive sonar images that employs a GA for determining detection thresholds.

There is a growing number of neural network applications at the intermediate vision level. An example is the work by Pomerleau [80] on network-based navigation of autonomous robots. Due to their inability to capture and generalize structural descriptions, NNs in general do not appear to be well suited for solving structural problems at the intermediate level. There are, however, certain functional mapping problems at the intermediate level that can be addressed successfully with NNs. For example, Aloimonos and Shulman [4] have suggested the use of NNs to learn the parameters involved in "Shape-from-X" problems.

Intermediate-level composite structures are commonly detected by either bottom-up grouping criteria (see above) or specified *a priori* as prototype patterns that are searched for in a goal-directed manner (e.g., [71]). The work reported by Segen [98] addresses some aspects of learning composite structural concepts from examples, however, no results have been shown on real images. Structural feature detection is usually based on a fixed set of visual primitives for which efficient detection algorithms are available. The incorporation of features of varying complexity has been addressed using only fixed, domain-independent grouping criteria. The problem of automatically forming intermediate-level perceptual shape concepts has found considerable attention in the psychological field recently.

9.5 Explanation-Based Learning

Explanation-based learning (EBL) [28] is an extension to an earlier concept called "explanation-based generalization" described by Mitchell et al. in [66]. Both paradigms are based on the same idea of using strong domain knowledge to "explain" why a given training example is a member of the concept being learned.

The domain knowledge (or domain *theory*) required in EBL consists of three main components:

1. A specification of the *types* and *properties* of the objects being dealt with.
2. A set of inference rules for inferring relations and properties from given relations and properties, and possible transformations between objects in the domain.
3. A library of problem-solving operators (schemata) that were either learned from earlier training examples or are hand coded.

The learning task in EBL can be stated as finding a generalized sequence of legal transformations (a schema) to derive the goal configuration from a given initial configuration. This is usually accomplished in a two-step process:

1. Construct an explanation that is causal with respect to the domain knowledge. This is similar to constructing a proof sequence for a theorem with respect to a set of axioms.
2. Generalize that explanation into a new schema by looking for the weakest preconditions under which the same explanation would apply.

The main limitation of EBL in its original form lies in the fact that the domain knowledge must be complete. If a given training example cannot be explained in terms of the existing domain knowledge, no generalization and thus no learning can take place. Another issue is the way the domain knowledge is specified and used. In "pure" EBL, the domain knowledge is expressed in the form of first-order logic predicates or Horn clauses, which provide no notion of proximity or similarity in a quantitative sense. However, many domains require handling of approximate, distorted, or noisy descriptions, and are thus not well suited for EBL in its original form. As a consequence, there have been several suggestions for extending the capabilities of EBL, in particular for relaxing the problem of incomplete and and possibly incorrect domain knowledge by combining analytical (EBL) and inductive learning [99, 67, 76, 103].

A second shortcoming of EBL is its strong dependence of a "good" encoding of the domain theory rules, which makes it difficult to design a domain theory that produces correct specializations. One approach for solving this problem is to employ a weaker semantic bias when searching for a solution path, which, however, requires the use of multiple training examples (EBL can, in principle, produce generalizations from single training examples) [37].

9.6 EBL and Visual Concepts

In this section, we describe the principles of applying EBL in the context of structural feature analysis and visual concept acquisition. The first step is to define the basic elements of the EBL paradigm, i.e., objects, relations, inference rules, initial state, and goal state in terms of the structural feature domain.

9.6.1 Elements of the Learning Problem

The primitives involved in this learning approach are two-dimensional geometric primitives. The assumption is that we have suitable mechanism available for extracting these primitives from images. Primitive classes include zero-dimensional primitives (points), one-dimensional primitives (straight line segments, arcs), and fully two-dimensional primitives (closed curves, elliptical regions, parametric blobs, etc.), as indicated in Figure 9.2. We call these three primitive classes $\mathcal{P}_0$, $\mathcal{P}_1$, and $\mathcal{P}_2$, respectively.

The domain knowledge in this case consists of

1. the properties of the individual primitives,
2. the spatial relations between primitives, and
3. a set of operators for combining (grouping) primitives into more complex arrangements.

The knowledge can be interpreted as a picture language (or algebra) for describing almost arbitrary configurations of picture primitives. In general, there is more than one possible description for a given arrangement of picture primitives. The *learning problem* consists of finding the simplest description (or a small *set* of simple descriptions) for a given picture configuration with respect to the current domain knowledge. The simplified descriptions found in the learning process become new intermediate-level visual concepts that are added to the current domain knowledge and can, in turn, become part of other object descriptions.

To evaluate the complexity of a particular description, each operator is associated with a cost term that represents the complexity of applying that operator or transformation. A similar approach is used in most approximate string matching techniques, where certain costs are associated with each character insertion, deletion, and replacement to compute a minimum "string edit" distance. The individual operator costs are assumed to be predefined and constant, at least originally. The questions of (a) how the operator costs should be related to the actual recognition mechanism and (b) if they can and should be learned pose interesting research topics.

9.7 Future Work

The work towards visual concept learning described in this chapter is still in an initial phase. Currently, our short-term goal in this problem area is to formalize the learning problem in precise terms and to specify suitable representations, learning algorithms, and performance measures. The plan is to adapt existing learning tools to this specific problem and to integrate these tools with other software components wherever possible. In addition, we are currently creating the necessary low-level operators for extracting structural features of various types that will allow to perform initial experiments on actual image data.

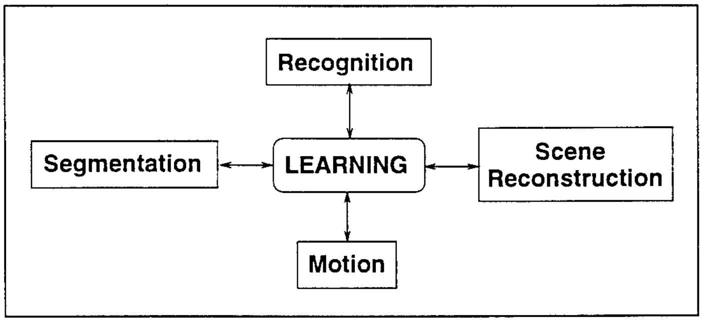
Chapter 10

A Learning System for Consolidated Recognition and Motion Analysis

A system for *l*earning *i*ntegrated *t*argeting and *e*xploration via *se*gmentation, *e*mplacement and *r*ecognition (LITE-SEER) is described. LITE-SEER uses learning augmented image understanding methodologies to identify and locate objects from a sequence of dynamic images for the following: (a) targeting and tracking in cluttered environments, (b) constraining object viewpoints for recognition, (c) detection of stationary and moving objects. To achieve these goals, the learning module (based on genetic and other algorithms) interacts cooperatively with the motion, segmentation and recognition modules. Experimental results on dynamic image sequences that detect, identify and locate obstacles like cones, cans, and wedge-shaped objects are presented.

10.1 Introduction

Tracking of moving and static objects, and exploration in an unknown or partially known environment are important applications of computer vision. Recognition and reconstruction of objects in a scene are often required for this purpose. Reconstruction of an object involves determining the shape of the object as well as the position and orientation of the object in



(a)

Figure 10.1: Overview of LITE-SEER.

3–D. The recognition of an object is of course limited by the range of models available in the model database; in practice the database could be extended by the acquisition of new models. Building an environmental model using depth information and models has been studied earlier [92]. However, most algorithms are still at an early stage and are not robust. For better performance of recognition algorithms the incorporation of depth information obtained from motion analysis can be of considerable help. Similarly, motion analysis can be assisted by the recognition of objects.

In the overview of the LITE-SEER system in Figure 10.1, the *motion module* determines dense depth maps from a sequence of 2–D images. The *segmentation module* can segment either 2–D intensity images or dense depth maps. The *recognition module* uses the pre-stored models and information from the segmentation and motion modules to recognize objects over multiple frames. The *learning module* is central to the system and allows for the cooperation between the segmentation, motion and recognition modules. The system attempts to secure the 3–D position of objects through motion algorithms and identify them via the recognition module. Then the objects can be placed in a 3–D map in the course of scene reconstruction. In short, the LITE-SEER project attempts to incorporate motion, segmentation and learning for model-based 3–D reconstruction and recognition from dynamic image sequences. LITE-SEER's application area includes outdoor navigation and robotics, target tracking, target recognition, surveillance etc. The fully implemented system is expected to run on the mobile vehicle *UCRover*, being developed at the University of California, Riverside. In this chapter, the focus is on the processes of motion analysis and recognition. They are combined to provide results that are better than those obtained from either of them separately. To achieve this goal segmentation is performed on the following:

(a) color images, (b) depth maps obtained via motion analysis. Segmentations of depth maps and intensity images are automated by using a computational learning paradigm to learn the parameters that have to be adjusted.

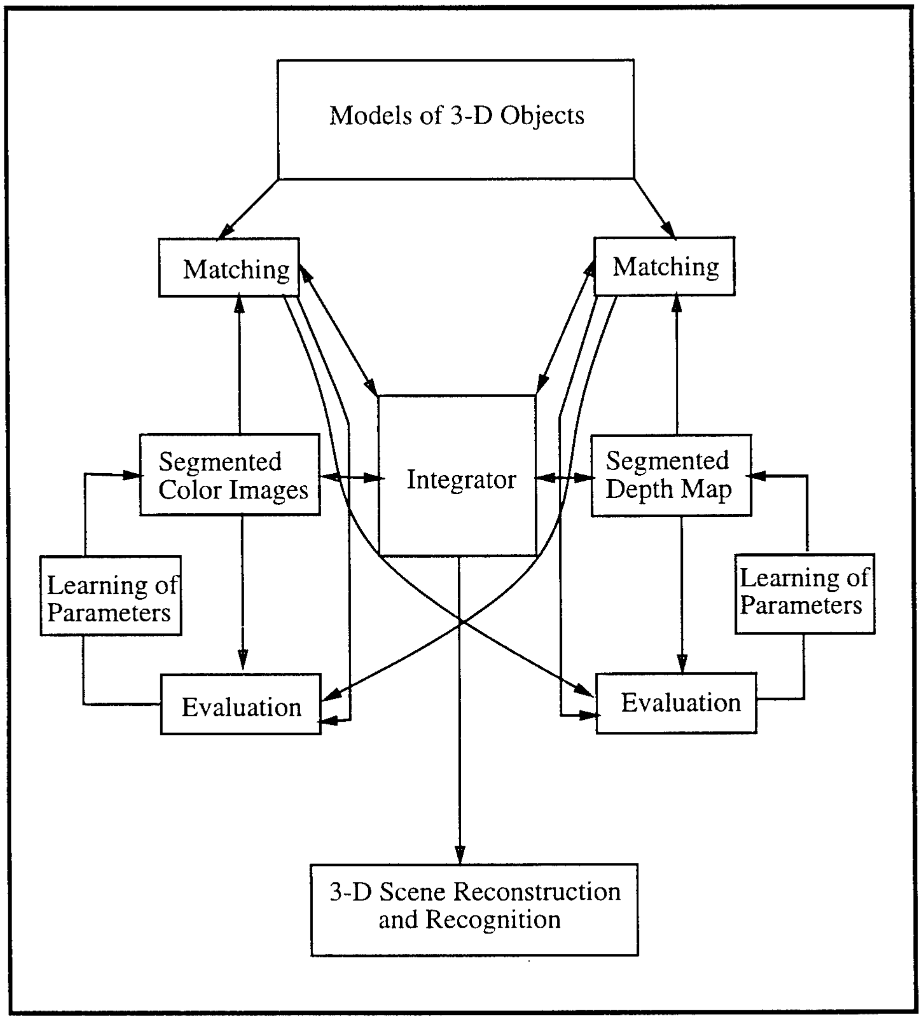
10.2 Components of LITE-SEER

The actual design and implementation of the LITE-SEER system is described in Figure 10.2. The pre-stored models of the objects are kept in a *model database.* The database contains CAD models of objects likely to be found in the 3–D scene. In the initial implementations of the work, the representations of the models are relatively straightforward and conform to simple geometrical figures and intensity characteristics of the objects when they are projected onto the image.

There are two separate channels which interchange information in the whole system: (a) Channel for the *segmentation of color images* (b) Channel for the *segmentation of depth images.* The two channels interact closely through the *Integrator* for recognition, 3–D position, orientation and shape determination of objects in the scene.

Even though the segmentation of images is an ill-posed problem in computer vision, for practical scenarios there is often no other alternative but to segment color images and depth maps. Usually this involves the adjustment of the values of several parameters for optimal segmentation. LITE-SEER uses *genetic algorithms* [15] to learn the optimal values of the parameters that have to be adjusted. Not only can genetic algorithms be used to learn the parameters for the segmentation of the depth maps and intensity imagery but they can also be extended to work in a similar way on infrared and LASAR imagery. Genetic algorithms have a high probability of locating the global optimum solution in a multidimensional search space. In addition, when multiple characteristics like depth and intensity are involved and the interactions among them are complex, the genes of the chromosomes can represent the various characteristics in the genetic algorithm.

As far as model-based segmentation of depth map is concerned, even though it is a well-studied subject [5], and algorithms have been implemented on parallel processors [21] the results obtained need improvement for robust performance in practical applications. For the purposes of this study, the method used for obtaining dense depth map is that described by Dutta in [34]. It derives dense depth maps from motion in both outdoor and indoor imagery to about 8% accuracy (at distances of up to 80 feet) in real-time in an SIMD mode of computation on the Image Understanding Architecture. The parameters for segmenting the depth map are determined through the application of genetic learning in a fashion similar to the determination of parameters for color segmentation.



(a)

Figure 10.2: Algorithmic components of LITE-SEER.

The two channels interact closely and influence the output of each other. The *evaluations of the fit* between the models and the objects in the segmented color and depth images are determined by considering the influence of the other channel, i.e., the color segmentation is influenced by the depth segmentation and vice-versa. The final output of the *Integrator* is

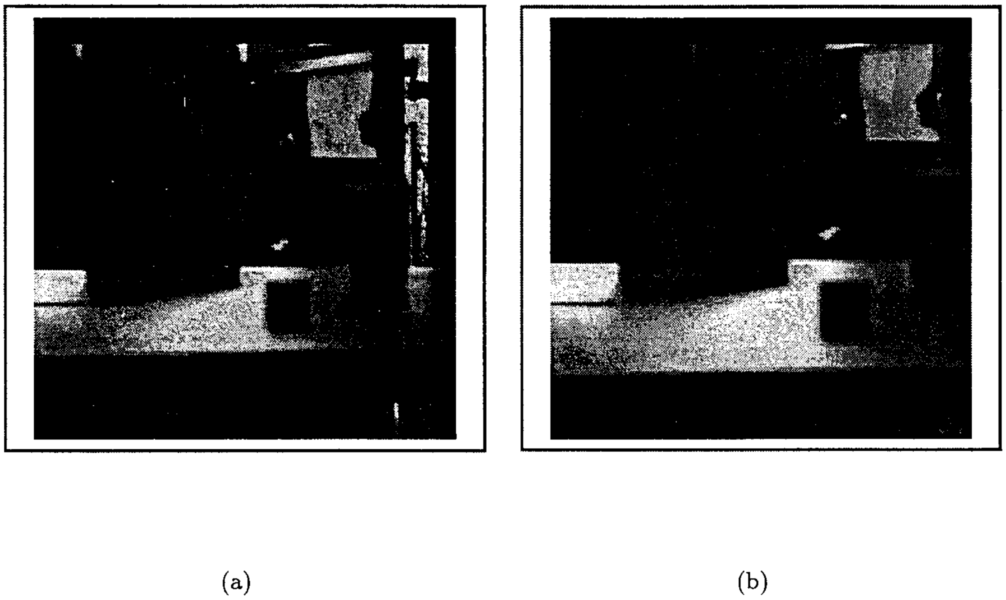


Figure 10.3: 1^{st} and 2^{nd} frames of a sixteen frame sequence.

the reconstructed 3–D scene with the recognized objects.

10.3 Experiments

Figures 10.3 and 10.4 show frames of two typical image sequences collected by moving cameras. The sequences will be referred to as "indoor" and "outdoor" sequence respectively. From the image sequences the motion module computes dense depth maps via the use of SIMD-based parallel depth from motion algorithms described in Dutta [34]. The dense depth map obtained for a subimage of the indoor sequence containing the "wedge" is shown in Figure 10.5. The dense depth maps obtained for two separate regions of the outdoor sequence containing cones and cans are shown in Figure 10.6. For the depth map of a subimage, the darker the gray scale the closer the environmental point is to the camera.

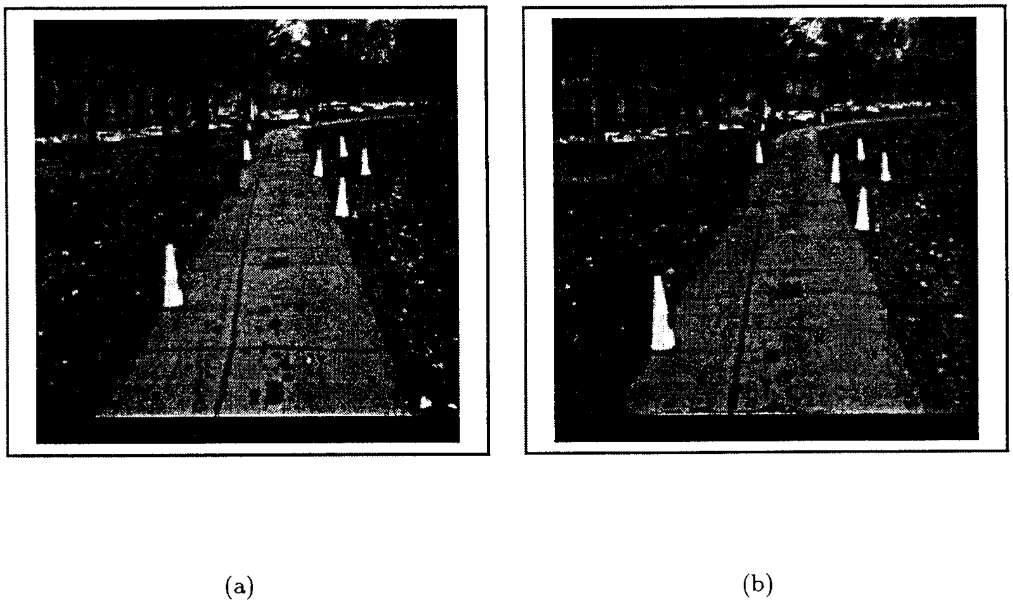


Figure 10.4: 1^{st} and 3^{rd} frames of a twenty frame sequence.

The result of intensity-based segmention of the "wedge" subimage of the indoor sequence is shown in Figure 10.7. The result of segmenting the central region of the first image of the outdoor sequence by using models of cones and cans based on intensity characteristics are shown in Figure 10.8. It can be seen that the "wedge," cones and can stand out clearly from the ground.

Genetic learning algorithms are used to apply the intensity segmentation results obtained in Figures 10.7 and 10.8 to segment the depth maps shown in Figures 10.5 and 10.6 such that the depth of the "wedge", cones and cans can be determined and they can be separated from the ground plane. With no other information other than depth maps complete separation of obstacles (e.g. cones, can) is not possible because *part* of the surroundings of an obstacle in the depth map is at almost the same depth as the obstacle and tends to merge with the foot of the obstacle (e.g. the depth of the bottom of cones and the surrounding ground is the same). This is different from an intensity image where all regions of an obstacle have

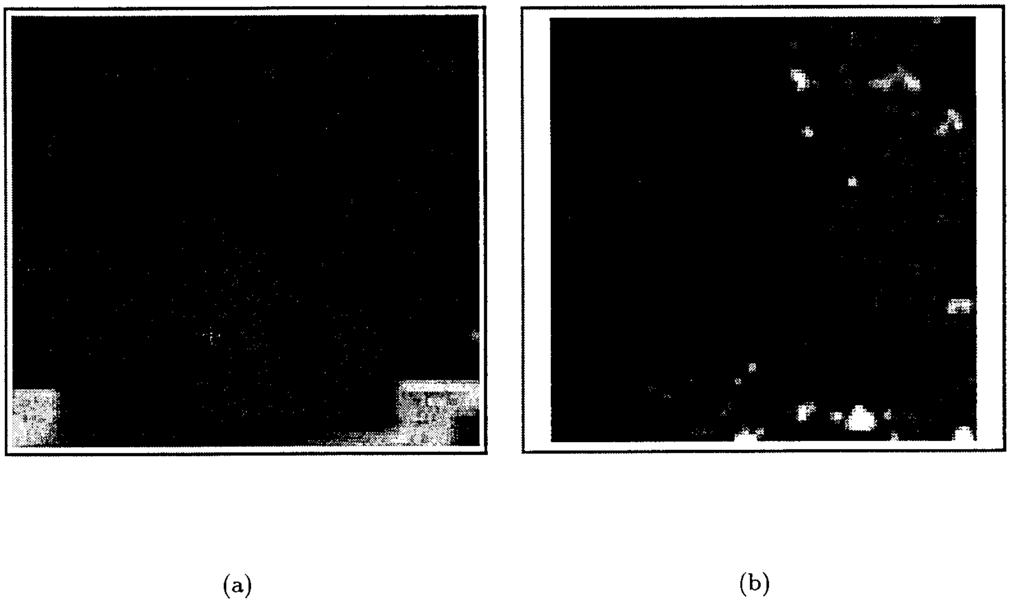
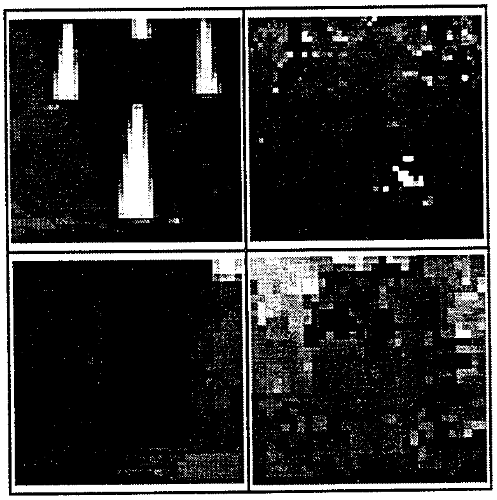


Figure 10.5: The subimage of the wedge of the indoor sequence is shown on the left and the dense depth map obtained from motion analysis is shown on the right hand picture.

a different image intensity from their surroundings. The genetic algorithm will try to learn the parameters for segmentation of the depth maps.

From experience it is determined that the "wedge" may be segmented from the depth map of the indoor image with two thresholds. Similarly the can may be segmented from the depth map of the outdoor image with two thresholds. However, several cones at varying distances require three thresholds. Hence, the genetic learning implementation for the automated learning procedure for the depth segmentation of "wedges" and cans have two genes corresponding to the two thresholds whereas for depth segmentation of the can there are three genes corresponding to the three thresholds.

The initial population of chromosomes (which contains the genes) is selected at random. Some information is known about the range of depths in the image and this is incorporated as a constaint while generating the random chromosomes. The goodness of each

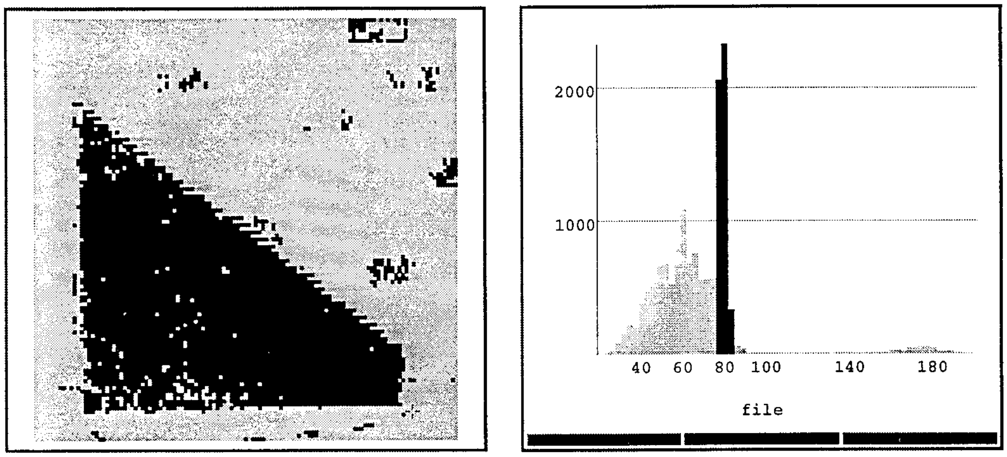


(a)

Figure 10.6: Cones and can from the outdoor sequence are shown on the left and their dense depth maps obtained purely from motion analysis are shown in the corresponding right hand pictures.

chromosome is then evaluated by an evaluation function $f(x_1, x_2)$ for the can and "wedge" images and $f(x_1, x_2, x_3)$ for the cone image where x_1, x_2 and x_3 represent the genes of the chromosome. The evaluation function tries to match the intensity segmented map with the hypothetically segmented depth map constructed from genes reflecting the thresholds. Succeeding generations of chromosomes are chosen applying a crossover rate of 0.6 and a mutation rate of 0.001. The number of generations created is limited by the processing time that can be used for the problem. The ten best parameter estimates for the thresholds of depth segmentation for the "wedge," cone, and cans are shown in Table 10.1.

The depth images corresponding to the results of Table 10.1 are illustrated in Figures 10.9 and 10.10. The average of the three best solutions were used for segmentation. Studies comparing the automatically generated results of Figure 10.10 with the manually obtained results show that the generated thresholds are close enough to the manually generated

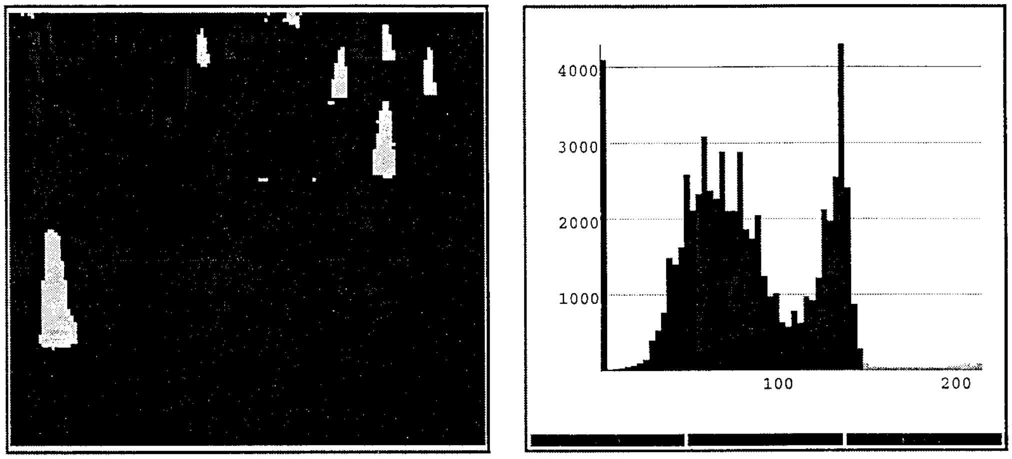


(a) (b)

Figure 10.7: Segmentation of the "wedge" based on intensity for the first image of the indoor sequence. The right picture is an histogram which illustrates the segmentation. The shading of the histogram corresponds to the shading on the segmented image. For example, the "wedge" has an intensity value between 76 and 84 and is shown in in black.

thresholds. Since the manually generated thresholds were chosen with great care after a lot of attempts, the automatically generated thresholds are excellent. The automatically generated thresholds are a lot better than any that are generated from default parameters.

Once the segmented depth map and the segmented intensity maps are obtained the scene can be reconstructed as shown in Figures 10.11 and 10.12. The triangular structure in the surface plot of Figure 10.11 is the "wedge." The three undulations in the surface plot of Figure 10.12 are the three cones. The depth of the "wedge," can and cones can be obtained from the histograms. The "wedge" is at a depth of 3 m. (10 feet), the can is at a depth of 46 feet, the near cone is at a depth of 36 feet and the farther cones are at a depth of 56 feet.



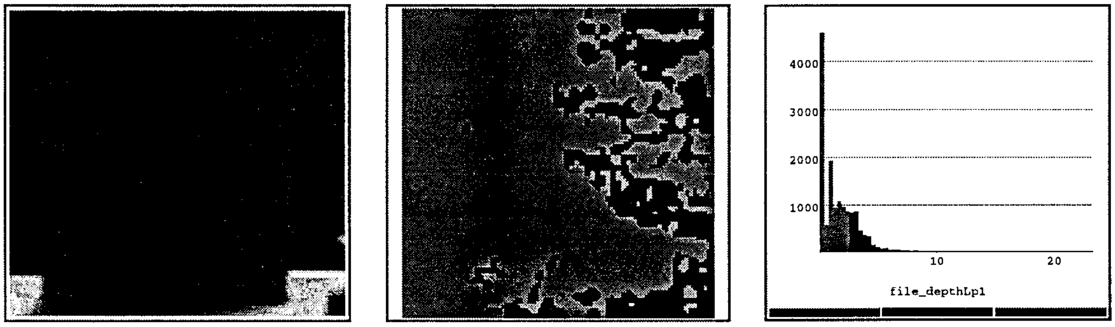
(a)

(b)

Figure 10.8: Segmentation of cans and cones based on intensity for the first image of the outdoor sequence. The cones have intensity greater than 148; the can has intensity less than 53; the intensity of the ground varies between 54 and 147.

10.4 Conclusions and Future Work

At the current state of the system, genetic learning has been used to segment the depth maps and combine them with intensity segmentation and models for 3–D reconstruction and recognition of simple objects. The future focus will be on the completion of learning and model-based algorithms to recognize and track 3–D objects for real applications.

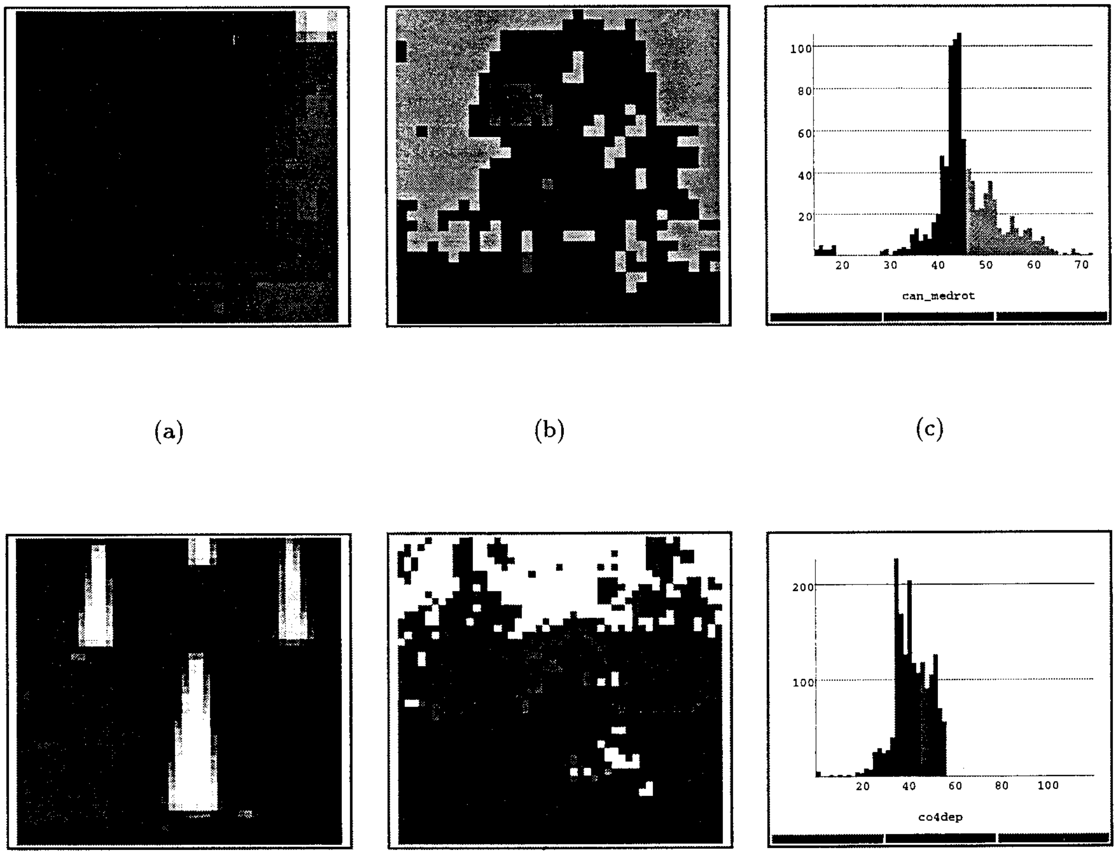


(a) (b) (c)

Figure 10.9: Depth segmentation of the "wedge" from genetic learning and motion analysis.

Table 10.1: The ten best solutions for thresholds with genetic learning for depth maps. t-1, t-2 and t-3 are the thresholds.

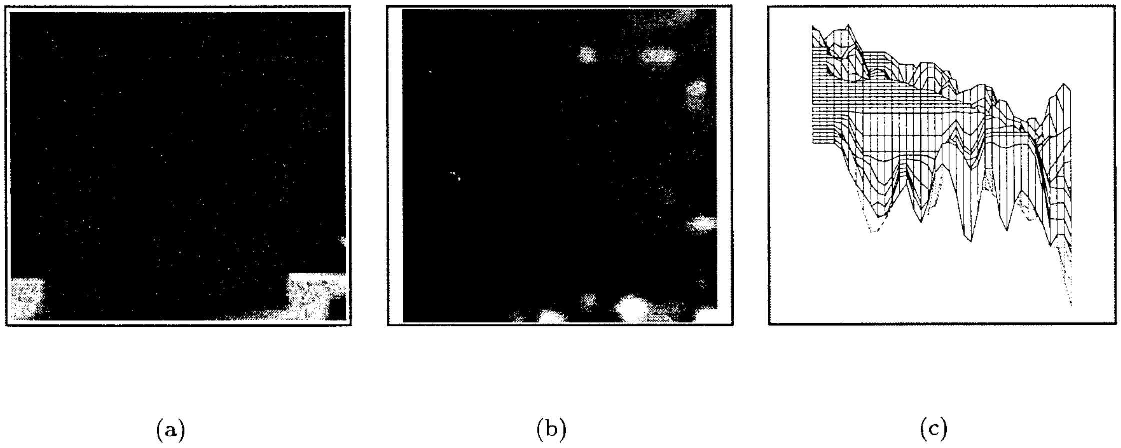
```
-----------------------------------
          For the ``wedge'' image
       t-1    t-2      evaluation
       0.38    3.19    6.1140e+03
       1.62    5.34    6.2830e+03
       0.71    3.37    6.1140e+03
       0.79    3.20    6.1140e+03
       1.94    5.34    6.2830e+03
       0.71    3.19    6.1140e+03
       0.79    3.19    6.1140e+03
       1.94    5.45    6.2830e+03
       0.79    3.37    6.1140e+03
       0.71    3.27    6.1140e+03

           For the can image
       t-1    t-2      evaluation
      32.11  45.32    9.8000e+01
      28.05  47.35    9.9000e+01
      31.10  47.35    9.9000e+01
      25.00  45.32    9.9000e+01
      31.10  45.32    9.7000e+01
      33.13  46.33    9.9000e+01
      26.02  45.32    9.9000e+01
      33.13  47.35    9.9000e+01
      33.13  45.32    9.7000e+01
      30.08  45.32    9.7000e+01

           For the cone image
   t-1    t-2    t-3      evaluation
  27.08  46.89  58.44    7.0300e+02
  24.60  46.89  57.62    6.9900e+02
  24.60  44.41  57.62    7.0100e+02
  27.08  43.59  57.62    6.8100e+02
  27.90  46.89  58.44    7.0300e+02
  27.08  44.41  57.62    6.8700e+02
  24.60  43.59  57.62    6.9500e+02
  27.90  43.59  58.44    6.9900e+02
  27.08  41.11  57.62    7.0100e+02
  27.08  46.89  57.62    6.8500e+02
-----------------------------------
```



(a) (b) (c)

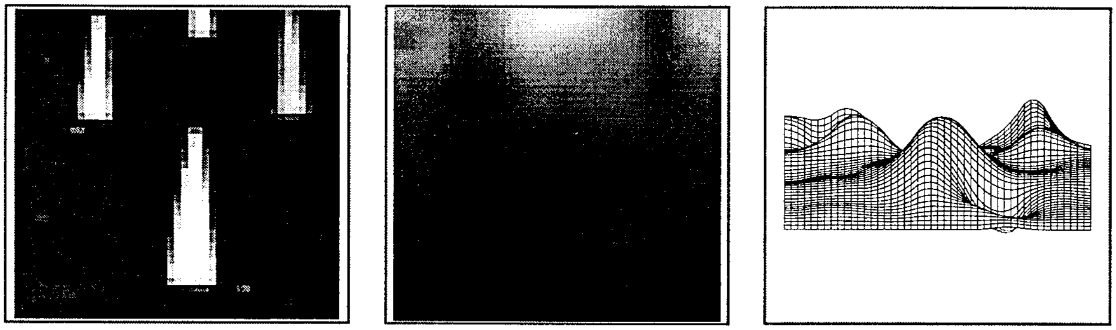
(d) (e) (f)

Figure 10.10: Depth segmentation of cans and cones from the depth maps obtained via genetic learning and motion analysis between frames 1 and 3.



(a) (b) (c)

Figure 10.11: Recognition and surface reconstruction of the "wedge." The middle picture is the smoothed depth map and the right picture is the reconstructed surface.



(a) (b) (c)

Figure 10.12: Recognition and surface reconstruction of the cones. The middle picture is the smoothed depth map and the right picture is the reconstructed surface.

Bibliography

[1] D. Ackley. Stochastic iterated genetic hillclimbing. Technical Report CMU-CS-87-107, Carnegie Mellon University, Dept. of Computer Science, March 1987.

[2] J.A. Adam. Peacekeeping by technical means. *IEEE Spectrum*, 23(7):42–56, July 1986.

[3] O.E. Agazzi and S.-S. Kuo. Hidden Markov model based optical character recognition in the presence of deterministic transformations. *Pattern Recognition*, 26(12):1813–1826, 1993.

[4] J. Aloimonos and D. Shulman. Learning shape computations. In *Proc. DARPA Image Understanding Workshop*, pages 862–873, 1987.

[5] F. Arman and J. K. Aggarwal. Model-based object recognition in dense depth images - a review. *ACM Computing Surveys*, pages 5–43, Mar. 1993.

[6] K. D. Ashley. Arguing by analogy in law: A case-based model. In D.H. Helman, editor, *Analogical Reasoning: Perspectives of Artificial Intelligence, Cognitive Science, and Philosophy.* Boston, MA: Kluwer, 1988.

[7] K. D. Ashley and E. Rissland. A case-based approach to modeling legal expertise. *IEEE Expert*, Summer 1988.

[8] T. Back, F. Hoffmeister, and H. P. Schwefel. A survey of evolution strategies. In *Fourth Intl. Conf. Genetic Algorithms*, pages 2–9, San Diego, Calif., July 1991.

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

[10] A. G. Barto. Reinforcement learning and adaptive critic methods. In David A. White and Donald Sofge, editors, *Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches*, pages 469–491. Van Nostrand Reinhold, 1992.

[11] B. Bhanu. Automatic target recognition: State of the art survey. In *IEEE Trans. Aerospace and Electronic Systems*, volume 22 of *AES*, pages 364–379, 1986.

[12] B. Bhanu, R.N. Braithwaite, W. Burger, S. Das, S. Rong, and X. Wu. Multistrategy learning for image understanding. Report to ARPA VISLAB-MSL-94-1, Univ. of California, Riverside, CA, Feb. 1994.

[13] B. Bhanu, B.L. Hutchings, and K.F. Smith. VLSI design and implementation of a real-time image segmentation processor. *International Journal of Machine Vision and Applications*, 3(1):21–44, January 1990.

[14] B. Bhanu and T. Jones. Image understanding research for automatic target recognition. In *DARPA Image Understanding Workshop*, pages 249–259, 1992.

[15] B. Bhanu and S. Lee. *Genetic Learning for Adaptive Image Segmentation.* Kluwer Academic Publishers, 1994.

[16] B. Bhanu and S. Lee. *Genetic Learning for Adaptive Image Segmentation.* Kluwer International Series in Engineering and Computer Science, Robotics: Vision, manipulation and Sensors (Takeo Kanade, editor). Kluwer Academic Publishers, Boston, MA, Spring 1994.

[17] B. Bhanu, S. Lee, and S. Das. Adaptive image segmentation using genetic and hybrid search methods. *IEEE Trans. on Aerospace and Electronic Systems*, July 1995.

[18] B. Bhanu, S. Lee, and J. Ming. Adaptive image segmentation using a genetic algorithm. *IEEE Trans. on Systems, Man and Cybernetics*, July 1995.

[19] B. Bhanu and J. Ming. Recognition of occluded objects: A cluster-structure algorithm. *Patter Recognition*, 20(2):119–211, 1987.

[20] B. Bhanu, J.C. Ming, and S. Lee. Closed-loop adaptive image segmentation. In *IEEE Conf. Computer Vision and Pattern Recognition*, pages 734–735, Maui, HI, 1991.

[21] B. Bhanu and L. A. Nuttall. Recognition of 3–D objects in range images using a butterfly multiprocessor. *Pattern Recognition*, 22(1):49–64, 1989.

[22] Bir Bhanu and Subhodev Das. *Computational Learning for Adaptive Computer Vision.* Plenum Publishing Company, 1994.

[23] A. Bhattacharyya. On a measure of divergence between two statisical populations defined by their probability distributions. *Bull. Calcutta Math. Soc.*, 35(3):99–110, 1943.

[24] R.E. Blahut. *Fast Algorithms for Digital Signal Processing.* Addison Wesley, 1985.

[25] K. W. Bower, L. O. Langley, B. Bhanu, and B. A. Draper. Report of the aaai fall symposium on machine learning and computer vision: what, why and how? In *Proceedings of the ARPA image Understanding Workshop*, pages 727–780, Monterey, California, November 1994.

[26] D. Chapman. Intermediate vision: Architecture implementation, and use. *Cognitive Science*, 16:491–537, 1992.

[27] R. T. Chin and C. R. Dyer. Model-based recognition in robot vision. *ACM Computing Surveys*, pages 67–108, March 1994.

[28] G.F. DeJong and R. Mooney. Explanation-based learning: An alternative view. *Machine Learning*, 1:145–176, 1986.

[29] K. DeJong. Adaptive system design: A genetic approach. *IEEE Trans. Systems, Man, and Cybernetics*, 10(9):566–574, 1980.

[30] K. DeJong. Learning with genetic algorithms: An overview. *Machine Learning*, 3:121–138, 1988.

[31] K. DeJong. Learning with genetic algorithms: An overview. *Machine Learning*, 3:121–138, 1988.

[32] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum-likelihood from incomplete data via the EM algorithm. *J. Royal Stat. Soc. Ser. B (methodological)*, 39:1–38, 1977.

[33] B. A. Draper, C. E. Brodley, and P. E. Utgoff. Goal-directed classification using linear machine decision trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(9):888–893, 1994.

[34] R. Dutta. *Depth from Motion and Stereo: Parallel and Sequential Algorithms, Robustness and Lower bounds.* PhD thesis, Department of Computer Science, University of Massachusetts at Amherst, September 1994.

[35] R. Dutta and B. Bhanu. A learning system for consolidated recognition and motion analysis. In *Proceedings of the ARPA Image Understanding Workshop*, pages 773–776, Monterey, California, November 1994.

[36] M. A. Fischler. On the representation of natural scenes. In A. R. Hanson and E. M. Riseman, editors, *Computer Vision Systems.* Academic, New York, 1978.

[37] N.S. Flann and T.G. Dietterich. A study of explanation-based methods for inductive learning. *Machine Learning*, 4:187–266, 1989.

[38] K. S. Fu. Stochastic automata as models of learning systems. In J. M. Mendel and King-Sun Fu, editors, *Adaptive, Learning, and Pattern Recognition Systems: Theory and Applications*, pages 393–431. New York: Academic Press, 1970.

[39] K. S. Fu and J. K. Mui. A survey on image segmentation. *Pattern Recognition*, 13:3–16, 1981.

[40] D. Genter. Structure-mapping: A theoretical framework for analogy. *Cognitive Science*, 7(2):411–436, 1983.

[41] A. M. Gillies. *Machine Learning Procedures for Generating Image Domain Feature Detectors.* PhD thesis, University of Michigan, Ann Arbor, MI, April 1985.

[42] D. Goldberg. *Computer-Aided Gas Pipeline Operation using Genetic Algorithms and Rule Learning.* PhD thesis, Dept. of Civil Engineering, University of Michigan, Ann Arbor, MI, 1983.

[43] D. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, Reading, Mass., 1989.

[44] D. E. Goldberg and Holland J. H. Special issue on genetic algorithms. *Machine Learning*, 2/3, 1988.

[45] D.E. Goldberg. Dynamic system control using rule learning and genetic algorithms. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 588–592, 1985.

[46] J. Grefenstette. Optimization of control parameters for genetic algorithms. *IEEE Trans. Systems, Man, and Cybernetics*, 16(1):122–128, January 1986.

[47] V. Gullapalli. A stochastic reinforcement learning algorithm for learning real-valued functions. *Neural Networks*, 3(6):671–692, 1990.

[48] V. Gullapalli. Associate reinforcement learning of real-valued functions. Technical report, Department of Computer and Information Science, University of Massachusetts, Amherst, May, 1993.

[49] P. A.V. Hall and G. R. Dowling. Approximate string matching. *acmcs*, 12(4):381–402, 1980.

[50] R.M. Haralick and L.G.. Shapiro. Image segmentation techniques. *Computer Vision, Graphics and Image Processing*, 29:100–132, 1985.

[51] Y. He and A. Kundu. 2-D shape classification using Hidden Markov Model. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(11):1172–1184, Nov. 1991.

[52] J. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, Mich., 1975.

[53] J. Holland. Escaping brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule-based systems. In R. Michalski, J. Carbonell, and T. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, volume II, pages 593–623. Morgan Kaufman, Los Altos, Calif., 1986.

[54] X.D. Huang, Y. Ariki, and M.A. Jack. *Hidden Markov Models for Speech Recognition.* Edinburgh, UK: Edinburgh University Press, 1990.

[55] J.J.Grefenstette, R.Gopal, B.J.Rosmaita, and D.Van Gucht. Genetic algorithm for the traveling salesman problem. In *Proc. of Intl. Conf. Genetic Algorithms and Their Applications*, pages 160–168, July 1987.

[56] D. W. Aha, D. Kibler and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66, 1991.

[57] J. L. Kolodner, R. L. Simpson, and K. Sycara. A process model of case-based reasoning in problem solving. In *Proc. 9th Intl. Joint Conf. Artificial Intelligence.* Los Angeles, CA, 1985.

[58] A. Kundu, Y. He, and P. Bahl. Recognition of handwritten word: First and second order hidden Markov model based approach. *Pattern Recognition*, 22(3):283–297, 1989.

[59] J.W. Sherman, D.N. Spector, C.W. "Ron" Swonger, L.G. Clark, E.G. Zelnio, M.J. Lahart, and T.L. Jones. Automatic target recognition systems. In L. Shumaker, editor, *The Infrared and Electro-optical Systems Handbook*, pages 343–402. SPIE Optical Engineering Press, 1993.

[60] K.I. Laws. The phoenix image segmentation system: Description and evaluation. Technical Report 289, SRI International, Dec. 1982.

[61] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1:541–551, 1989.

[62] M.D. Levine and A.M. Nazif. Low level image segmentation: An expert system. *IEEE Trans. Pattern Analysis and Machine Intelligence*, PAMI-6:555–578, Sept. 1984.

[63] D.G. Lowe. Three-dimensional object recognition from single two-dimensional images. *Artificial Intell.*, 31:355–395, 1987.

[64] V.R. Mandava, J.M.Fitzpatrick, and D.R. Pickens III. Adaptive search space scaling in digital image registration. *IEEE Trans. on Medical Imaging*, 8(3):251–262, 1989.

[65] T. M. Mitchell. The need for bias in learning generalizations. Technical Report CBN-TR-117, Department of Computer Science, Rutgers University, 1986.

[66] T.M. Mitchell, R.M. Keller, and S.T. Kedar-Cabelli. Explanation-based generalization: A unifying view. *Machine Learning*, 1:47–80, 1986.

[67] T.M. Mitchell and A.B. Thrun. Explanation-based learning: A comparison of symbolic and neural network approaches. In *Machine Learning: Proc. of the Tenth International Conference*, pages 197–204, Amherst, MA, June 1993. Morgan Kaufmann.

[68] S. Mitra and S. K. Pal. Self-organizing neural network as a fuzzy classifier. *IEEE Trans. Systems, Man, and Cybernetics*, 24(3):385–399, 1994.

[69] D.J. Montana. Empirical learning using rule threshold optimization for detection of events in synthetic images. *Machine Learning*, 5:427–450, 1990.

[70] A. W. Moore and C. G. Atkeson. Memory-based reinforcement learning: Converging with less data and les real time. In J. H. Connell and S. Mahadevan, editors, *Robot Learning*, pages 79–103. Kluwer Academic, 1993.

[71] T.N. Mudge, J.L. Turney, and R.A. Volz. Automatic generation of salient features for the recognition of partially occluded parts. *Robotica*, 5:117–127, 1987.

[72] K. S. Narendra and M. A. L. Thathatchar. *Learning Automata: An Introduction.* Prentice Hall, Englewood Cliffs NJ, 1989.

[73] 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.

[74] P. Fua P. Suetens and A. J. Hanson. Computational strategies for object recognition. *ACM Computing Surveys*, 24(1):5–59, 1992.

[75] P.W. Pachowicz. Integrating low-level features computation with inductive learning techniques for texture recognition. *Intl. J. Pattern Recognition and Artificial Intelligence*, 4(2):147–165, 1990.

[76] M.J. Pazzani. Detecting and correcting errors of omission after explanation-based learning. In *Proc. IJCAI-89*, pages 713–718, 1989.

[77] J. Peng. *Efficient Dynamic Programming-Based Learning for Control.* PhD thesis, Northeastern University, Boston, MA, 1993.

[78] J. Peng and B. Bhanu. Closed-loop object recognition using reinforcement learning. In *Proc. DARPA Image Understanding Workshop*, pages 777–780, Monterey, CA, 1994. November 14-16.

[79] J. Peng and R. J. Williams. Incremental multi-step q-learning. In *Proceedings of the 11th International Conference on Machine Learning*, pages 226–232, New Brunswick, NJ, 1994.

[80] D.A. Pomerleau. Neural network based autonomous navigation. In C. Thorpe, editor, *Vision and Navigation: The Carnegie Mellon Navlab*, pages 83–93. Boston, MA: Kluwer, 1990.

[81] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[82] K. Price R. Ohlander and D. R. Reddy. Picture segmentation using a recursive region splitting method. *Computer Graphics and Image Processing*, 8:313–333, 1978.

[83] L.R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[84] L.R. Rabiner and B.H. Juang. An introduction to hidden Markov models. *IEEE ASSP Magazine*, pages 4–16, January 1986.

[85] B. Ravichandran. 2D and 3D model-base matching using a minimun representation criterion and hybrid genetic algorithm. Technical Report 105, Center for intelligent robotic systems for space exploration, Rensselaer Polytechnic Institute, Troy, New York, 1993.

[86] G. Reynolds and J.R. Beveridge. Searching for geometric structure in images of natural scenes. In *DARPA IU Workshop*, pages 257–271, Los Angeles, CA, 1987.

[87] W.R. Reynolds. Toward quantifying infrared clutter. *Proceedings SPIE*, 1311:232–240, 1990.

[88] C. Riesbeck and R. Schank. *Inside Case-Based Reasoning.* Hillsdale, NJ: Erlbaum, 1989.

[89] S. Rong and B. Bhanu. Characterizing natural backgrounds for target detection. In *Proc. ARPA Image Understanding Workshop*, Monterey, California, November 14-16 1994.

[90] G. Roth and M. D. Levine. A genetic algorithm for primitive extraction. In *Proc. Intl. Conf. Genetic Algorithms*, pages 487–494, San Diego, Calif., July 1991.

[91] G. Roth and M. D. Levine. Geometric primitive extraction using a genetic algorithm. In *Conf. on Computer Vision and Pattern Recognition*, pages 640–643, Champaign, IL, June 1992.

[92] Y. Roth-Tahak and R. Jain. Building an environment model using depth information. *IEEE Computer*, pages 85–90, June 1989.

[93] Hannan Samet. The quadtree and related hierarchical data structure. *ACM Computing Surveys*, pages 187–260, 1984.

[94] E. Saund. Symbolic construction of a 2-d scale-space image. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 12(8):355–395, 1990.

[95] J. Schaffer and J. Grefenstette. Multi-objective learning via genetic algorithms. In *Proc. 9th Intl. Joint Conf. Artificial Intelligence*, pages 593–595, Los Angeles, Calif., 1985.

[96] R. Schank and R. Abelson. *Scripts, Plans, Goals, and Understanding.* Hillsdale, NJ: Lawrence Erlbaum, 1977.

[97] Roger Schank. *Dynamic Memory: A Theory of Learning in Computers and People.* Cambridge, MA: Cambridge University Press, 1982.

[98] J. Segen. Learning structural descriptions of shape. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 96–99, 1985.

[99] A. M. Segre, editor. *Workshop on Combining Empirical and Explanation-Based Learning*, Proc. 6th International Workshop on Machine Learning, pp. 1–93, San Mateo, CA, 1989. Morgan Kaufmann.

[100] A. Sha'ashua and S. Ullman. Structural saliency: The detection of globally salient structures using a locally connected network. In *Proc. IEEE Secor.d Intl. Conf. Comp. Vision*, pages 321–327, Tarpon Springs, FL, Dec. 1988.

[101] S. Shafer and T. Kanade. Recursive region segmentation by analysis of histograms. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 1166–1171, 1982.

[102] S. Shafer and T. Kanade. Recursive region segmentation by analysis of histograms. In *IEEE INternational Conference on Accoustics, Speech, and Signal Processing*, pages 1166–1171, 1982.

[103] J.W. Shavlik and G.G. Towell. An approach to combining explanation-based and neural learning algorithms. *Connection Science*, 1(3):231–253, 1989.

[104] C.Y. Suen. *n*-gram statistics for natural language understanding and text processing. *IEEE Trans. Pattern Analysis and Machine Intelligence*, PAMI-1(2):164–172, 1979.

[105] R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

[106] A.J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Information Theory*, 13(2):260–269, 1987.

[107] C.J. Watkins. *Learning from Delayed Rewards.* PhD thesis, King's College, University of Cambridge, Cambridge, MA, May 1989.

[108] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 3:229–256, 1992.

[109] R. J. Williams and J. Peng. Function optimatization using connectionist reinforcement learning. *Connection Science*, 3(3), 1991.

[110] P.H. Winston. *Artificial Intelligence.* Reading, MA: Addison-Wesley, 1984.

[111] E.J. Yannakoudakis, I. Tsomokos, and P.J. Hutton. *n*-grams and their implication to natural language understanding. *Pattern Recognition*, 23(5):509–528, 1990.

[112] E. G. Zelnio. Importance of sensor models to model-based vision. In H. N. Nasr, editor, *Automatic Object Recognition*, pages 112–121. Bellingham, WA: SPIE Optical Engineering Press, 1991.