A KNOWLEDGE REPRESENTATION SYSTEM FOR HUMMINGBIRD FORAGING BEHAVIOUR IN A LABORATORY ENVIRONMENT

by

PETER CAHOON

B.Sc., Dalhousie University, 1970.

M.A.Sc., University of British Columbia, 1978.

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

in

THE FACULTY OF GRADUATE STUDIES

Interdisciplinary Studies

We accept this thesis as conforming

to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA

May, 1987

In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of Interdisciplinary Studies

The University of British Columbia
1956 Main Mall
Vancouver, Canada
V6T 1Y3

Date April 29 /97

Abstract

A knowledge representation system is presented for studying hummingbird behaviour in a laboratory environment. It is shown that a set of procedural rules can be developed, based on numerical, symbolic and heuristic techniques to aid in discovering how a hummingbird learns a simple spatial grid pattern of food sources. The use of different types of tree-like data structures facilitates a systematic representation of the knowledge fragments and allows a thorough cross-examination of both the experimental designs and the hypotheses. The problem of analyzing a non-uniformly sampled time series of behaviour observations is discussed and a solution proposed that uses a mathematical matching algorithm called warping. A trajectory of individual feeder visitations is generated by a bird behaviour model. The technique of warping is used to test if this trajectory can be mapped to another generated by a bird foraging. The two-dimensional analogue of the warping technique is applied to the spatial grid in order to evaluate the degree of spatial specialization in the bird's foraging behaviour. A correlation measure is applied to groups of pairs of rule combinations to ascertain which of these account for most of the observed behaviour. It is shown that by using a collection of different types of similarity measures a procedural approach can be formulated to aid in the representation of the knowledge accumulated by a hummingbird during the course of a spatially distributed foraging experiment. These procedures are arranged in a hierarchy of choices and implemented in an interpreter which formed the basis for an expert system in hummingbird spatial foraging. Experimental applications of these numerical algorithms and data structures are presented. The system was then tested on a complete series of behaviours by testing five different individuals on the same design. The procedural algorithms were calibrated on the first individual and then applied to subsequent individuals to test the knowledge representation derived from the first case. The results from this experiment suggest that a knowledge representation system composed from these rule fragments can be developed into a grammar that would standardize the testing of all spatial foraging experiments. In addition it is indicated that representing knowledge as a hierarchy of procedural options is of use in testing the way in which experimental knowledge is gathered. The implications of this knowledge of spatial foraging can be tested interactively as an experiment progresses.

Table of Contents

Abstract	ii
Table of Contents	iii
List of Figures	vii
List of Copyrighted Products	x
Knowledge Representation in a Biological Environment	xi
Acknowledgements	xii
Chapter 1	
1.1 Introduction	1
1.2 The Importance of Spatial Pattern Learning in Hummingbird Foraging	6
1.3 The Importance of Representation Systems that can Organize, Remember, and Learn	7
1.4 Elements of a Language for Investigating Hummingbird Spatial Learning	8
1.5 Experimental Design	9
1.6 Summary	13
Chapter 2	
2.1 Perception, Recognition and Heuristic Searches	14
2.2 Methodology for Recognizing Patterns in a Data Base	20
2.3 Summary	25
Chapter 3	
3.1 Combining a Knowledge Base and an Interpreter to Form an Expert System	26
3.2 Including Scripts in the Knowledge Base of the Expert System	34
3.3 Coupling the Expert System with the Numerical Software	35

3.4 Modelling Irregularly Sampled Spatial Sequences in One Dimension 36

3.5 Partitioning a Foraging Pattern while Conserving Adjacencies 38

3.6 Warping Irregularly Sampled Sequences in One Dimension 39

3.7 Warping Irregularly Sampled Sequences in Two Dimensions 43

3.8 Searching for Rules in a Data Base 45

3.9 Summary 47

Chapter 4

4.1 Implementing the Knowledge Representation System and Pattern Matching Algorithms on a Micro-Computer 48

4.2 System Hardware Design 48

4.3 System Control Design 49

4.4 Using Graphics Windows to Run Scripts 49

4.5 Using Scripts in a Graphics Window to Animate Behaviour Patterns 51

4.6 Creating Rule Stereotypes to Assess Learning Patterns 57

4.7 Creating Rules to Learn Other Rules 61

4.8 Searching for Movement Combinations Using a Grammar 61

4.9 Context Free Grammars 62

4.10 Applying a DCG and Movement Rules to Foraging Sequences 64

4.11 Searching for Minimal Sets of Rule Combinations to Form a Searching Strategy 65

4.12 Some Selective Differences in Learning an Experimental Design 68

4.13 Summary 84

Chapter 5

5.1 Learning Spatial Patterning Using a Knowledge Base 85

5.2 Grouping Rule Preferences in a Propositional Lattice 89

5.3 Correlation Measures on Time Series of Rule Frequency Pairs 91

5.4 Analysis of the Variation in Use of Propositional Pairs 99

5.5 Warping of Trajectories in One Dimension onto a Target Trajectory 112

5.6 Summary 116

Chapter 6

6.1 Summary and Conclusions 117

6.2 Concurrency and Spatial Learning 119

6.3 Changes Required in the System Design 120

6.4 Knowledge Representation in Biological Systems 121

Appendix 1

1.1.1 Warping of Data Sequences that have been Irregularly Sampled in Time and Space 122

1.1.2 One Dimensional Time Warping 122

1.1.3 Design of a Cost Function 123

1.1.4 The Correlation Coefficient Q 124

1.1.5 Constraints and the Penalty Function 124

1.1.6 Summary of 1.1 127

1.2.1 Two dimensional Spatial Warping 128

1.2.2 Non-Uniform Sampling in Two Spatial Dimensions 128

1.2.3 Theorem 1.2.1.
The Uniform Two Dimensional Sampling Theorem 129

1.2.4 Mapping onto a Hexagonal Lattice 130

1.2.5 Theorem 1.2.2
The Non-Uniform Two Dimensional Sampling Theorem 131

1.2.6 Constructing a Generalized Hexagonal Tessellation 132

1.2.7 Adjacency Conserving Partition Mapping 136

1.2.8 Generalized Hexagonal Tessellation 136

1.2.9 Theorem 1.2.3 Invertibility of a Mapping 137

1.2.10 Implementation of the Two Dimensional Transform 139

1.2.11 Summary of 1.2 141

References 142

List of Figures

Chapter 1

Figure 1.0. Experimental design 4

Figure 1.1. An illustration of what has been done in this thesis 10

Figure 1.2. Experimental room 11

Chapter 2

Figure 2.0. An illustration of a generic hummingbird frame 15

Figure 2.1. A foraging sequence frame 17

Figure 2.2. An illustration of an inverse composition hierarchy 19

Figure 2.3. A new experiment frame 21

Figure 2.4. A knowledge representation composition hierarchy 23

Chapter 3

Figure 3.0. An illustration of possible model movements 37

Figure 3.1. An illustration of the feeder numbering 40

Figure 3.2. An illustration of a one dimensional warping function 41

Figure 3.3. An illustration of warping applied to two behaviour trajectories 42

Chapter 4

Figure 4.0. An illustration of the software flow of control 50

Figure 4.1. An illustration of how the animation script works 52

Figure 4.2(a through l). Animation of bird 1's foraging in 10 trial blocks for all three experimental days 53-55

Figure 4.3(a). Levels of structured 'simple' learning 58

Figure 4.3(b). Levels of structured 'complex' learning 58

Figure 4.3(c). A script of mixed searching fragments 58

Figure 4.3(d). An illustration of a 'good' strategy of diagonal search 67

Figure 4.4(a through h). Animation of bird 2's foraging in 10 trial blocks for all three experimental days 69-70

Figure 4.5(a through e). Percent success for each trial, for each day for each bird 71-73

Figure 4.6(a through k). Animation of bird 3's foraging in 10 trial blocks for all three experimental days 75-77

Figure 4.7(a through h). Animation of bird 4's foraging in 10 trial blocks for all three experimental days 78-79

Figure 4.8(a through j). Animation of bird 5's foraging in 10 trial blocks for all three experimental days 80-82

Chapter 5

Figure 5.0. A propositional lattice 87

Figure 5.1. Reference numbers for the rule subset 91

Figure 5.2. Matrix pair elements for the correlation matrix 92

Figure 5.3. An illustration of how to read figures 5.3(a) through 5.7(c) 93

Figure 5.3(a,b,c). Total correlation for all propositional pairs for bird 1 on days 1, 2, and 3 respectively 95-96

Figure 5.4(a,b,c). Total correlation for all propositional pairs for bird 2 on days 1, 2, and 3 respectively 97-98

Figure 5.5(a,b,c). Total correlation for all propositional pairs for bird 3 on days 1, 2, and 3 respectively 100-101

Figure 5.6(a,b,c). Total corrleation for all propositional pairs for bird 4 on days 1, 2, and 3 respectively 102-103

Figure 5.7(a,b,c). Total correlation for all propositional pairs for bird 5 on days 1, 2, and 3 respectively 104-105

Figure 5.8. An illustration of how the time series were generated 107

Figures 5.8(a through g). Cross Spectra for various propositions and various birds 108-111

Figure 5.9(a).One dimensional warping of bird 1's trajectory onto a 'good-search' target trajectory for day 3 113

Figure 5.9(b). One dimensional warping of bird 3's trajectory onto a 'good-search' target trajectory for day 3 113

Figure 5.9(c). One dimensional warping of bird 5's trajectory onto a 'good-search' target trajectory for day 3 114

Appendix 1

Figure 1.2(a). An illustration of two dimensional warping 133

Figure 1.2(b). An illustration of the spatial warping search strategy 135

Figure 1.2(c). A raw 2-dimensional grid of a foraging sequence 140

Figure 1.2(d). The same grid after warping 140

Copyrighted Products

The following copyrighted product names were used in this thesis

APPLE is trademark of the Apple Corporation

Arity is a trademark of Arity Corporation

AT is a trademark of IBM Corporation

AutoCAD is a trademark of Autodesk Corporation

DEC,PDP, and VAX are trademarks of Digital Corporation

Enhanced Graphics Adapter is a trademark of IBM Corproation

HP and Hewlett-Packard are trademarks of Hewlett Pacakrd Corporation

Lotus is a trademark of Lotus Corporation

MS-DOS is a trademark of Microsoft Corporation

Symbolics is a trademark of Symbolics Corporation

Sysdyne is a trademark of Computerland Corporation

Tecmar is a trademark of Tecmar Corporation

Transputer is a trademark of INMOS Corporation

UNIX is a trademark of Bell Laboratories

Xenix is a trademark of IBM Corporation

Knowledge Representation in a Biological Environment.

"Environments are invisible.
Their ground rules, pervasive structure,
and overall patterns elude easy perception."

Marshall McLuhan,
The Medium is the Massage.

"Take a line divided into two unequal parts,
one to represent the visible order, the other
the intelligible, and divide each part again
in the same proportion ... "

Plato,
Republic, Book VI, section 509.

"Remember, no matter where you go, there you are."

Buckaroo Banzai,
The Adventures of Buckaroo Banzai
Across the Eighth Dimension.

Acknowledgements

I would like to thank my committee, Bob Blake, Lee Gass, Bill Havens, Peter Lawrence, Tad Ulrych and Carl Walters for their academic support.

I would also like to thank H. R.(Mac) MacCarthy for his tireless efforts in helping to assemble and edit this thesis.

Thanks to Bob Shore and Ben Dover for helping put this computing system together, and to D. MacCarthy for assistance with the hardware programming, including complete instructions as to how one de-bounces a read signal.

Finally, a special note of thanks to Busan, whose ventral spirit and bottomless pockets supported this impossible project.

Chapter 1

1.1 Introduction

Environments are not passive structures. It can be argued that the ways in which organisms arrange and adapt their behaviour to a changing environment and how they use contextual information, may ultimately determine their success or failure. This contextual information should ideally be smoothly integrated by the organism into a representation system of appropriate responses to a large variety of situations.

Foraging research, in general, is directed towards analyzing an organism's behaviour in an extended time frame. Typically, this time frame is measured in days or weeks (Krebs, 1983, Pyke 1984, Mangel, 1986, McNamara, 1986). This thesis investigates how a hummingbird, *Selasphorous rufus*, adapts its foraging responses to a spatial array of feeders, figure 1.0, over a short time period(trial to trial); in addition it investigates the ways in which the spatial-temporal aspects of the birds' learning can be represented formally in terms of rules of action and facts in a data base.

The experiment analyzed in this thesis concerns itself with what has been learned during a 3 day period that has been broken up into 40, 40 second trials each day. During this period the spatial pattern learning by 5 different hummingbirds was studied in order to clarify the extent to which these bird's spatial memory was an important factor in determining where they foraged. The experiment was designed and run by Sutherland (1986) as part of his investigation of hummingbird spatial learning.

There is one key point in the search for the types of rules to be included in this data base. The key feature is the requirement that the experimenter test the ways that he or she represents what is thought to be true about spatial learning in this laboratory environment against the bird's actual behaviour. This means that there are in fact two representation systems to be tested.

The first representation system is an assumed one. The assumption is that the birds have an internal representation system of expectations about the allocation of the filled feeders. This assumed system gradually becomes more defined as the experiment unfolds.

The second representation system is the one the experimenter composes by observing the bird's efforts to learn the spatial array. From the observation process, the experimenter decides how to

code this information into a set of movement rules and search types that will ultimately compose the basis of a knowledge representation system for foraging in a laboratory environment.

The rule database must consequently evolve from the environment in which it occurs. This approach differs widely from the normal manner in which foraging behaviour is studied. The first difference is that the amount learned is not measured as a percent change in success rate (Sutherland, 1986). Learning is defined systematically as a collection of spatial patterns and trajectories and the manner in which they change. Although the success rate is presented it is of secondary interest. The primary interest is the development of a representation system that changes as the birds learn. This systematic coding of what is known about spatial learning into rules and actions on those rules forms the basis of a knowledge representation system. This systematic formulation is in fact a technique of suspended judgment on the part of an experimenter, because the body of accumulated knowledge fragments is continuously tested by the organism's behaviour. This interactive testing and revision on the part of an experimenter clearly defines the failures and gaps in the way one chooses to view a complex system whose components are context dependent.

The main purpose in developing a representation system having these features was to answer the following questions:

1. How do hummingbirds learn spatial patterns?

2. How does the experimenter learn how hummingbirds learn spatial patterns?

3. How does one integrate these two questions into a representation system that can solve both of these problems at the same time?

To aid in this investigation, a representation system was constructed containing three different levels of patterns. These patterns were made up of enumerated types of movements that were observed as the birds foraged on the grid shown in figure 1.0.

The first level of pattern representation was composed of those groups of movements which, when aggregated, produced some coarse level partitioning of the grid. Typically, this coarse level pattern would delineate whether the boundaries of the foraging search were either on the upper or lower half of the grid, or in some quadrant.

The second level of pattern representation was composed of groups of contiguous movements that could be labelled as proceeding in a particular direction on the grid. These allowed a finer level of representation of patterns as right and left across rows, up and down columns, and up and down diagonals.

The third level of pattern representation was composed of combinations of the first and second level patterns. Since the number of ways they can be arranged is large, specific pattern recognizers in the form of computer algorithms were developed to aid in recognizing when they occurred.

Pattern recognition of this type presents a complex problem. Aside from the combinatoral search problems, the pattern combinations themselves do not appear in a precise order and tend to overlap one another. Consequently, the way in which the experimenter sets up his or her systematic representation system is important. What is stated to be known in terms of observable patterns stored in a database, and which particular set of foraging responses the bird is using in a given situation, usually do not match exactly. Complex patterns of behaviour are not immediately visible to observers. They tend to evolve first as fragments, composed of smaller, inter-connected patterns, and then become obvious through repetition. On a longer time scale they eventually form some general spatial pattern of foraging behaviour. An additional level of complexity is added because the experimenter does not know at the beginning of the experiment what these general patterns will be.

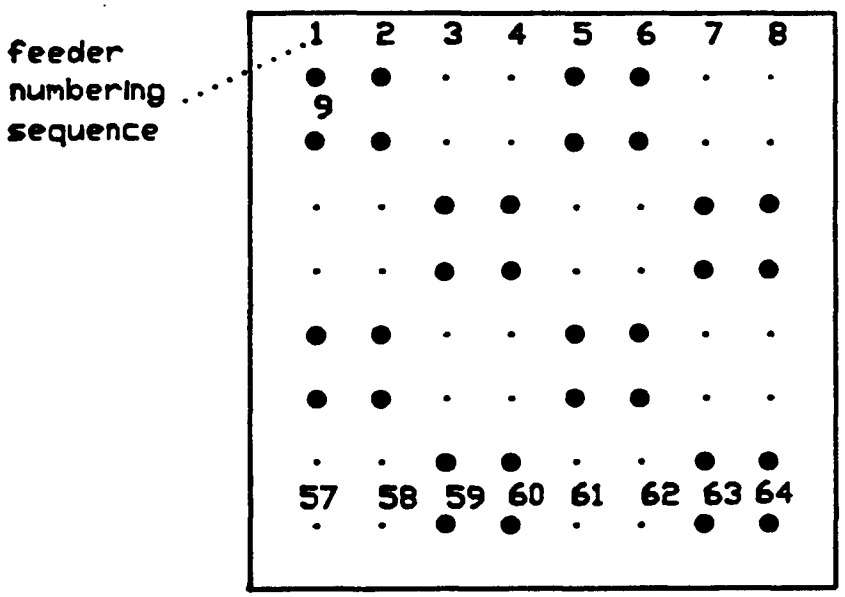
When developing pattern matching algorithms the coding of the kind of evidence that makes up a particular type of pattern must be as unambiguous as possible. The quest for what should be considered acceptable as unambiguous evidence of a pattern ultimately tests the way in which the experimenter represents this composite body of patterns, actions and consequences.

Learning processes are not, in general, directly observable. Their nature and component structure are generally inferred indirectly by observing changes in behaviour patterns. Kamil and Yoerg (1982), concluded that it is difficult to tell whether an observed difference in behaviour is due to the fact that different reasoning processes are at work, or whether the same processes are being used, but appear different because of differences in experimental setting.

A system of knowledge representation, such as the one to be developed in this thesis, requires constant revision by the experimenter. What at onset appears to be the simplest manner in which

Experimental design

An illustration of the filled feeder array



June 1985

figure 1.0

to represent the bird's spatial learning is continuously tested by the organism's behaviour. This interactive testing and revision on the part of an experimenter clearly defines the failures and gaps in the way one chooses to view a foraging environment's intrinsic texture, and serves to clarify the course of action to be taken.

Gass and Montgomerie(1981) summarized the types of questions that must be contained in any formalized knowledge representation that proposes to investigate short term behavioural adaptations. These are as follows:

1. "Perception. What kinds of information are animals sensitive to, how sensitive are they, and how does their perception affect their problem solving abilities?

2. Decision making. How much and what kind of information do animals use in decision making, and how do their information processing skills affect their performance?

3. Learning. How do animal's problem-solving abilities change with their experience?"

This thesis deals with nearly all aspects of these general questions. Experiments are generally designed to test some specific hypothesis. The methodology in this thesis is designed to allow multiple hypotheses to exist and be tested interactively during any part of the ongoing experiment. In the experiment to be investigated here, the hummingbird was observed foraging in a laboratory environment, and controlled only by its adaptation to the spatial pattern of feeders. The design consisted of an array of equally spaced nodes on a rectangular grid. Each node had a feeder that could be filled with a fixed amount of sugar water. An experiment consisted of filling half of the feeders in a pre-determined spatial pattern. The hummingbird was then presented with this pattern and allowed to forage for a pre- determined length of time. All of the bird's movements were recorded in each of 40 trials. For experimental details see section 1.5 and Sutherland(1986).

The computational problem is to track the bird's progress through a foraging bout, using various types of numerical and pattern recognition algorithms in order to detect the three levels of spatial patterns. One wishes to combine the results of this tracking in a way that allows a structured, interactive manipulation of the patterns in order to discover further clues to the types of algorithms that a hummingbird uses for spatial pattern recognition. All of the hypotheses to be tested must be coded procedurally as part of an integrated system that works in a goal-directed manner.

1.2 The Importance of Spatial Pattern Learning in Hummingbird Foraging

Spatial pattern learning has been established as an important part of how hummingbirds use learning and memory while foraging (Sutherland 1986). In his thesis, Sutherland implements experimental designs of the type shown in figure 1.0, plus several other arrangements of varying complexity. He concludes that it was important that these birds first develop a coarse grained expectation about the quality of groups of feeders. It was demonstrated that the birds did not depend on simple area-restricted search mechanisms to perform well in simple or complex spatial arrays. It was also shown that the birds developed expectations about the quality of feeder locations. This was true even when the spatial patterns were complex. The birds consistently used the regularities in the patterning of the feeders when foraging.

Sutherland concluded that "if regularity of spatial patterning is an important general determinant of spatial memory capacity and learning rates, then identifying how animals recognize these patterns becomes important in specifying the constraints on animal memory for spatial locations."

Pattern recognition, and the representation of patterns in a systematic manner lies in the domain of Artificial Intelligence(AI). Implementing a knowledge representation system involves using AI pattern classification techniques, as well as the development of procedures that allow directed, interactive manipulation of these classifications in order to make inferences about pattern groupings.

The particular contribution of the present thesis in studying the role of spatial pattern learning by these hummingbirds, is the ability to ask the computer programs **HOW** a particular pattern was discovered by its procedures. This means that at any level of investigation, a complete history of the patterns or groups of patterns used in the detection process can be printed out and studied in detail. Through the use of a continually updated data base, a complete history of every pattern found can be stored and used as an active part of the search for more complex patterns. The resulting system serves as a prototype that can be applied, in a modular form, to a variety of similar control structures, with very little internal change.

1.3 The Importance of Representation Systems That Can Organize, Remember and Learn

Effective foraging requires that animals acquire information about spatial patterns of food sources quickly. This information must be highly organized and remembered for use at a later time (Krebs et al, 1983). The environments in which these food sources are found exist on different spatial scales. Hummingbirds forage over a range of meadows, specialize in particular sections of these meadows, then within patches, and finally within flower types. Other animals use spatial information to guide foraging efforts. Gould (1986), showed that honey bees use landmarks to guide their way to feeding locations and also to guide their way back to the hive.

Models of foraging, however, leave much to be desired when accounting for **WHAT**, and **HOW** patterns are organized and remembered. Mangel and Clark(1986) have presented what is stated to be "towards a unified foraging theory". For Mangel and Clark, an adequate model for foraging must have three things:

1. A set of state variables characterizing the physiological state of the forager that change in a stochastic fashion to reflect environmental changes and the consequences of decision making.

2. Some mathematical concept of fitness.

3. A foraging strategy in the form of a Markov process embedded in a dynamic programming algorithm. This algorithm would optimize the individual's fitness over a long time interval.

Mangel and Clark's model explicitly states that it will ignore learning. Remembering is not a contextual function, but rather a set of probabilities stored from previous state transitions and assumes perfect recall of all previous events. However, selective remembering, forgetting and learning, have been identified as key requirements for an efficient forager(Krebs et al, 1983).

Shettleworth (1983) in a review article observed that a Clark's nutcracker cached 33,000 pine seeds, in groups of 4 or 5 during the months of September through November. These caches were covered with snow until the end of March. The bird, however, returned and uncovered thousands of these caches throughout the winter. Shettleworth also reviewed another experiment that was conducted in a laboratory. A dozen tree branches were drilled with a total of 100 holes, and mounted in stands on the floor. A marsh tit was allowed to store 12 seeds wherever it wanted. It was then

removed from the lab and returned 2.5 hrs later. It recovered 10 out of 12 seeds but inspected only 24 holes. It is clear from this series of experiments, as well as from a wealth of other descriptive foraging studies, that successful foragers are not using a simple stochastic optimization to find these items. They effectively learn where things are located, remember them for long periods, and have well defined expectations as to which combination of items are required to satisfy their needs.

Any collection of algorithms that is composed with the purpose of studying spatial foraging behaviour must be able to **organize**, **remember** and **learn** on a variety of spatial and temporal scales: Gould and Marler, (1987).

1.4 Elements of a Language for Investigating Hummingbird Spatial Learning

There are two component factors in implementing a spatial pattern representation system. The first is the representation of the patterns themselves in a data base. The second is allowing access to the procedural representation, by the experimenter, to take place in a natural manner. Natural language processing is a principle focus of artifical intelligence research. The necessity of developing a common language for referring to and connecting ideas has been discussed at great length by many authors. For a comprehensive review, see Winograd (1982). Chomsky, (1963), outlined the formal properties that grammars for a language should contain.

Ideally one would like to develop a complete language, that is both descriptive and analytical in format. This language would connect the procedural representation of how the bird learns with the experimenter's perception of how things should go together in his or her representation system. It would represent what is known to the experimenter about hummingbird spatial learning in a laboratory environment. The way people communicate their representations of ideas to one another is through a natural language, such as English. This hummingbird language would communicate, through common units, how the experimenter chooses to represent individual cases, such as the experiment studied in this thesis. The units of this language must form a hierarchy in the same way as an English sentence is a hierarchy of clauses, groups, and words. The structure of these individual hierarchial units is composed of one or two simple units filling a particular role and connected to the units immediately below it. They would be classified by the roles to be filled at the level above. These would finally be collected into a system of choices representing the options open to the experimenter.

This system of choices and actions forms a network for interpreting and classifying not only the 3 levels of patterns, but also the appropriate analytical methods to be applied. The elements of this thesis only crudely approximate the complete set of components that would be required to define this language, but they do appear to work well in this restricted environment and they do provide a prototype for further development of this concept.

Figure 1.1 illustrates the elements of this system as it was structured in this thesis. At the top most level, numbered 1, is a natural language interface that uses a definite clause grammar(DCG). This permits an English interface of possible choices and options available to the experimenter. The second level of implementation, indicated by the large bracket, indicates the reasoning components of the system that enable it to organize, remember and learn new information from a data base. It is also designed to answer queries from the top most level and supply options for furthering the experimenter's investigations. The bottom two elements, numbered 3 and 4, are specific to a particular problem and a specific experiment, respectively. The specific problem has been outlined in the first section of this chapter, and the specific experiment is outlined in the next section. The key components are contained in the top two levels. The expert system and component parts are problem independent; that is, their reasoning mechanisms, organization, and learning capabilities are not changed by changing the problem. The reasoning mechanisms simply expand upon one level of complexity to incorporate another. This is similar to the manner in which a language evolves as new information, words, and concepts are acquired. The implementation of these features, plus the pattern recognition implementation in a micro-computer environment is the subject of the rest of this thesis.

1.5 Experimental Design

All of the data generated for analysis by these numeric and symbolic methods result from an experimental design illustrated in figure 1.2, and discussed in Sutherland (1986). The grid was mounted as shown in an enclosed room that was fitted with an array of 64 single-dose feeders on the wall. The room also had a one-way mirror for observing the bird. The locations to be filled are shown at "g". They were allocated in a checkerboard design, as illustrated in figure 1.0.

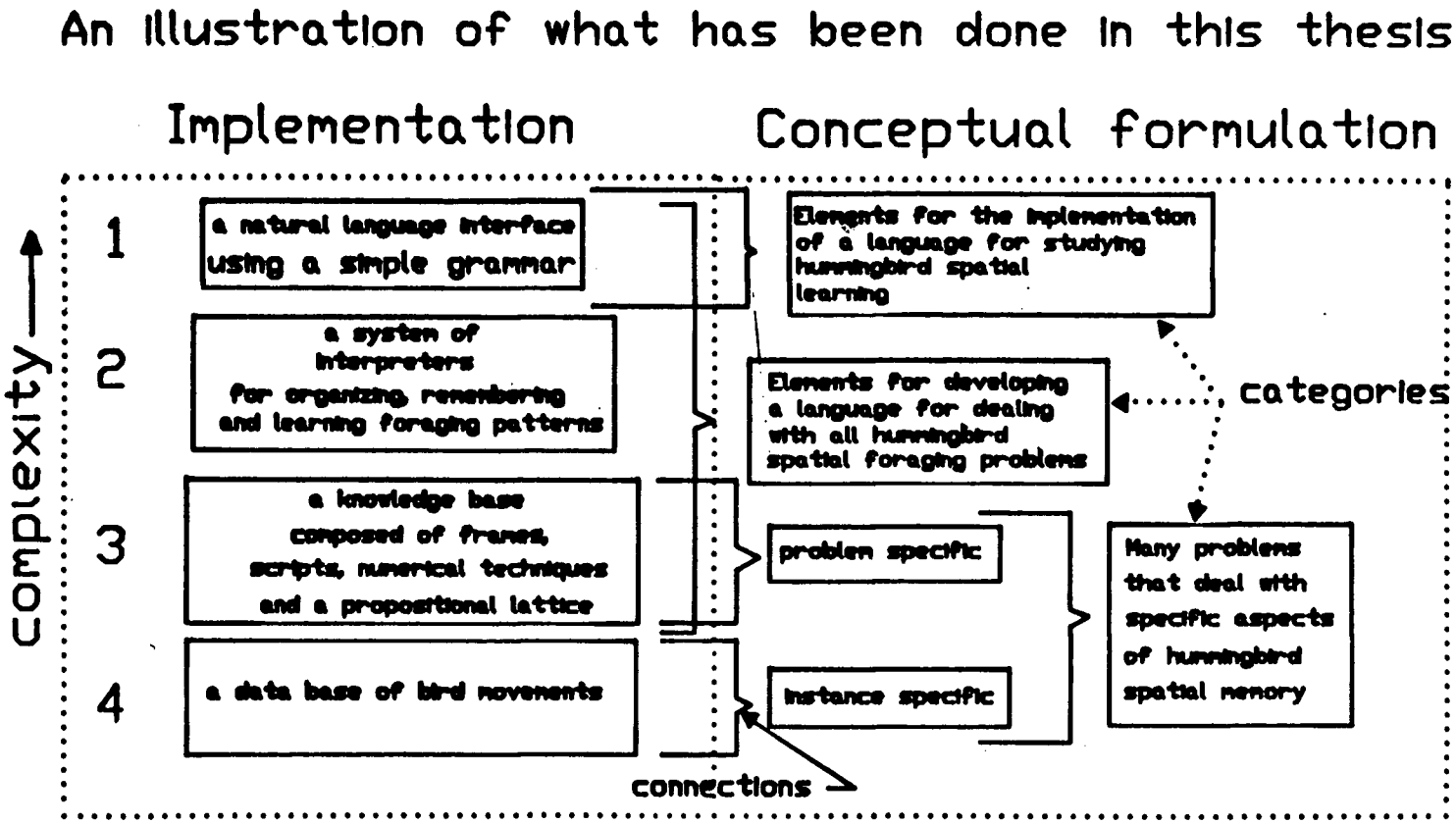

An illustration of what has been done in this thesis
Implementation
Conceptual formulation
complexity
1
2
3
4
a natural language interface using a simple grammar
a system of interpreters for organizing, remembering and learning foraging patterns
a knowledge base composed of frames, scripts, numerical techniques and a propositional lattice
a data base of bird movements
Elements for the implementation of a language for studying hummingbird spatial learning
Elements for developing a language for dealing with all hummingbird spatial foraging problems
problem specific
instance specific
categories
Many problems that deal with specific aspects of hummingbird spatial memory
connections

figure 1.1

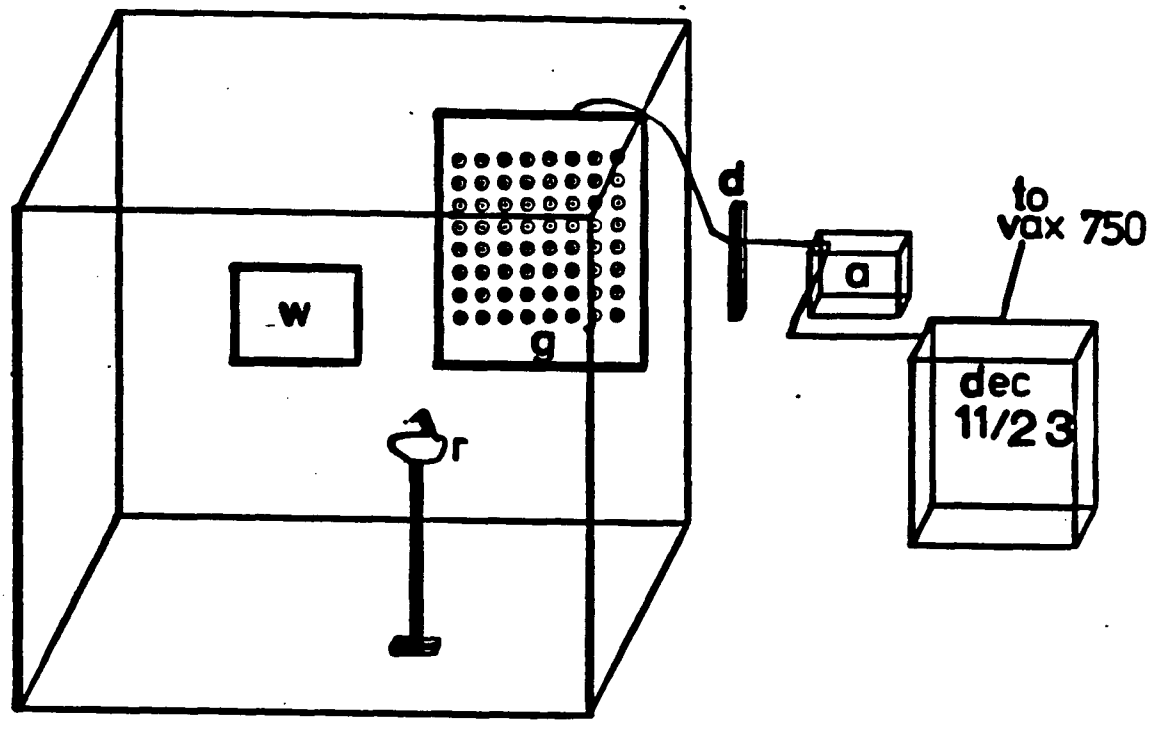

Experimental room
d
to
vax 750
a
w
g
dec
11/23
r
a = Apple II
d = decoder
g = grid
r = wire ring
w = window

figure 1.2

The number of loaded feeders for any trial was 32 and remained constant throughout the three-day experiment for all of the five hummingbird subjects. The feeding array was made from small hollow tubes attached to individual reservoirs containing a 15% concentration of sugar in water. Each tube with its reservoir was attached to a base which in turn was attached to a metal panel having a regular array of holes. Each feeder hole contained a photodarlington photocell which was triggered by the shadow cast by the bird's beak as it entered the hole. Each feeder reservoir contained $2\mu l$ of sugar water. The assembled feeders were attached so as to be even with the mounting board and had no markings to distinguish one from another. Prior to the commencement of a trial the array was covered with a window blind. This blind was subsequently raised and the foraging allowed to continue for 40 s.

Each trial started with the bird positioned as in figure 1.2, perched on a wire ring labelled "r" that was supported by a wooden stand. Each probe of the bird's beak into a feeder location was conveyed *via* a decoder "d" and then to the APPLE II computer labelled as "a". The location was converted to integer-numbered units. The information was then conveyed serially to a Dec PDP 11/23 and formatted. Next, this information was down-loaded to a VAX 750 main frame computer and reprocessed and reformatted into a portable format. Finally, having recoded all of the data in acceptable format, it was transferred to an IBM AT micro-computer where the rest of the system development and analysis took place.

1.6 Summary of Chapter 1.

-The problem of knowledge representation in foraging behaviour was introduced.

-The importance of spatial pattern learning in hummingbird foraging was presented.

-The importance of having a knowledge representation system that can remember, organize, and learn spatial patterns was presented.

-Examples of the ability of birds to remember the location of thousands of food items months after caching them were presented.

-The need for a procedural language for studying hummingbird spatial learning was presented.

-The experimental design was presented.

Chapter 2

2.1 Perception, Recognition and Heuristic searches.

In this thesis the knowledge representation system uses frames and scripts as its principle data types. Frames were originally proposed by Minsky(1975), as a basis for understanding visual perception. Scripts which are frame-like structures specifically designed for representing sequences of events were proposed by (Schank and Abelson 1977). The two of these together are methods for organizing what is known about the bird's foraging patterns in a data base and they direct attention to the specific aspects of the knowledge that facilitate recall and learning; for further details concerning frames see Bobrow and Winograd(1977).

Frames provide a structure that permits newly received data to be interpreted through patterns acquired during previous experiences. The organization of this knowledge about these patterns facilitates expectation driven processing. This means that one looks for things based on the context where one thinks they should appear. The representational mechanism that makes this possible is called a slot. This is the place where the newly acquired pattern knowledge is fitted within the larger context provided by the frame.

A frame is composed of declarative and procedural information in pre-defined internal relations. Thus, a generic frame for a hummingbird would have knowledge hooks, or slots, for facts that are typically known about hummingbirds.

A hummingbird-frame would look like figure(2.0) to the biologist attempting to represent what is known about hummingbirds in general compared to a specific case. The second frame immediately underneath the hummingbird frame is labelled the bird_in_hand frame. The differences between the two frames are not differences of structure, but rather those of specificity.

Another feature of the frame data structure is its ability to determine whether it is applicable to a given situation. The underlying idea is that a likely frame is selected to aid in understanding the current situation (dialogue, scene, problem) and this frame in turn tries to match itself to the data it encounters; if it finds that it is not applicable, it can transfer control to a more appropriate frame. The knowledge used for the expert system in this thesis will be of three types:

An illustration of a generic hummingbird frame

A frame representation of a generic hummingbird

Generic hummingbird	
Slots	Values
Animal	bird
Genus	*Selasphorous*
Species	*rufus*
Sex	male, female
Time of year	winter, summer
Back coloring	brown, green; default is brown

An illustration of a bird_in_hand frame

A bird_in_hand frame

Bird_in_hand	
Slots	Values
Animal	bird
Genus	*Selasphorous*
Species	*rufus*
Sex	male
Time of year	winter
Back coloring	brown

figure 2.0

Firstly, each frame contains factual knowledge about the concept that the frame represents; for example, the bird-in-hand frame. This knowledge is represented as follows: declaratively, slots and values, and procedurally, as computations associated with slots. Thus data-driven and procedural computations are carried out together.

Secondly, each frame contains heuristic knowledge to guide the search process that is trying to satisfy the frame's concept. For example, has there been enough change in the searching in a trial to indicate that a different pattern is being employed?

Thirdly, frames form relationships with other frames creating hierarchical networks. This concept of a frame as it applies to tracking a laboratory foraging sequence, is illustrated in figure 2.1 as the foraging-sequence frame. The mnemonics **TD** and **BU** stand for top down and bottom up searches that execute procedures. A top down search is a hypothesis-driven search for a pattern or goal; it works both procedurally and numerically to try and satisfy the frame's concept. A bottom up search is driven by data and is referred to as a **part_of** relationship with the data and a procedure. The top down search radiates downward in a hierarchy and has a **this_has_parts** relationship with the data.

The frame illustrated in figure 2.1 consists of a set of named relations, each containing a name, a pointer to another frame, or an expectation indicating what type of information can be used to fill the slot or named relation. When an instance of a frame has been found it is copied into a list of frame instances, along with the slots particular to this structure. At the beginning of a sequence these slots will contain expectations, which are numerical thresholds for visitation and searching sequence, as well as their corresponding parameters. As the data are received, these expectations are altered procedurally, forming new instances and frames during the course of the experiment.

This foraging-sequence frame illustrates several of the features required to build a recognition model. In the foraging-sequence frame, one sees that the first slot of the frame contains the name of this stereotype frame, enabling one to refer back, at a later time, to this particular frame by using a pointer, or simply by using its name. The next 5 slots in this frame representation contain all that is known at this time about a foraging sequence. Recognition of this frame requires that the compositional relationships contained in this particular stereotype are found to be true. Thus, one searches for a spatial stereotype in the representation of the visitation frequencies that will, as

Frame: Foraging sequence		TD means top down search BU means bottom up search
Slots	Values	Procedures
perch	duration, # of times	TD find a flying terminator BU found a flying terminator
flying	foraging	TD find a perch BU found a perch .
move	up, down, diagonal, etc.	TD find a jab terminator BU found a jab terminator
jab	location, duration	TD connect to all preceeding events BU connected to all preceeding events
is_a	list of instances	connect to all previous instances

figure 2.1

quickly as possible, allow the recognition of the type of searching that this hummingbird is doing. Therefore, a particular type of foraging sequence is recognized by the discovery of its component parts and in addition, these component parts are structured in such a way that they represent a frame for a foraging-sequence.

Complex stereotypical concepts are represented by frames which are composed of other concepts represented by sub-frames. The resulting hierarchy structure is called a composition hierarchy. This static hierarchy represents the composition of all possible instances of this class of foraging-sequence frame. A foraging-sequence is composed of flying, which can be of two types: foraging, or non-foraging and is also composed of perching, moving, or probing. Each of these generic components forms a stereotypical class of objects which in turn is represented as a stereotype frame. Each one of these components consists of its own generic components which are represented by stereotype sub-frames. Figure 2.2 is an illustration of a composition hierarchy. This illustration demonstrates the inverse composition relationship between frames. Each stereotype has a compositional relationship with one or more sub-frames. In turn, each of these sub-frames has an inverse relationship with the whole frame. This relationship is known as the part_of relationship that is needed for the bottom up searches within the network of frames. The top down, or compositional relationship, moves down a hierarchy of frames.

Perception is a recognition task (Havens 1978), that comprises a description of a perceived concept from a sequence of external observations obtained, in this case, from monitoring a foraging bout. The experimenter's concept of what is required for perception is embodied in the frames, which are in turn composed of relationships among other, more primitive concepts or data. This set consists of several frames, each of which is capable of recognizing its own stereotypical concept. Thus each frame is a recognizer in a group of recognizers, so that at any time each frame has the active information necessary for the recognition of its concept, anywhere in the foraging bout. Active, heuristic knowledge such as this is called a method.

Methods, procedures that are tailored specifically to recognize their associated frames, permit the use of search techniques that are quite specific to their own domain. An example of a method is the use of a correlation measure in aiding the recognition of both the type of search and its preferred direction. These are particularly advantageous because one does not have to activate all frames to

An illustration of a composition hierarchy

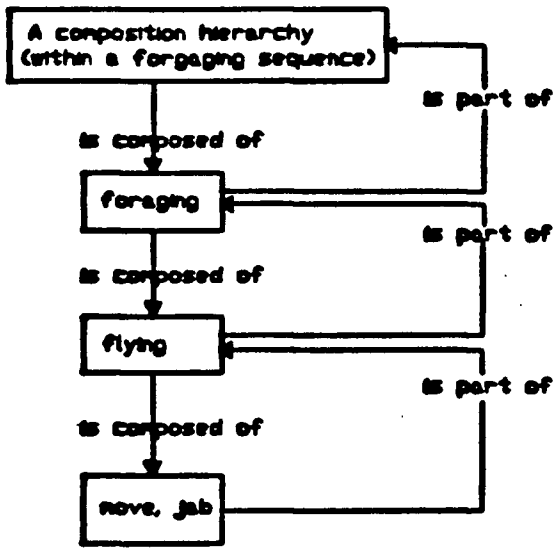


figure 2.2

locate some specific aspect that is of immediate use. The important point to note here is that the recognition procedure frame is quite different from the blind hypothesis type of searching where a particular hierarchical instance remains active until satisfied. The procedure is data driven and selects instances upon the discovery of some evidence that matches a particular expectation. This set of routines must be pre-specified before the data base is searched. These specifications consist of the set of named relations forming a hierarchy of activities that contain the start and end levels of named relations, or slots to be actively searched for in the data base. Frames could be formed as instances of those already defined at the beginning of a foraging bout, one would need only to provide the rules and attach them to the initial definitions. Thus, when this particular instance is satisfied it immediately becomes part of the original sequence.

2.2 Methodology for Recognizing Patterns in a Data Base

An example of the type of frame used in this representation system can be seen in figure 2.3. The procedural knowledge formalizes the expected form of hummingbird behaviour patterns into an array of types of search and the movements that are to searched for in a data base. The **new_experiment** frame was run at the commencement of experiment, or whenever the experimenter deemed it appropriate.

Note that each of the named actions, for example, **simple**, point to a procedure **tests** that is run to establish whether the type of searching used by the bird is in fact **simple**. There are several such tests, one for each search type. Each test has algorithms that examine groups of bird movements to establish their particular fact. The inheritence shown in figure 2.3 was implemented through the use of pointers that make up the command structure in the computer programming language prolog (Kowalski,1982).

Movements about the grid are detected when the bird probes a feeder hole. When the experiment begins, the search type has been set to the default, which is **simple**. If this search type should fail, because a boundary is detected, the rest of the collection of search types are tried until one succeeds. This structure is a simple tree that is descended, or traversed depending on which of the four search types succeeds. The slots can contain frames too. The **structured** model for searching can contain further specifications for what types of movements to test for in order for it to be more

Frame: New Experiment		TD means top down search BU means bottom up search	
Slots	Values	Procedures	
simple frame	procedural tests to satisfy a random search	TD	find a simple search pattern
		BU	found a simple search pattern
structured frame	procedural tests to satisfy a structured search	TD	find a structured search pattern
		BU	found a structured search pattern
complex frame	procedural tests to satisfy a complex search	TD	find a complex search pattern
		BU	found a complex search pattern
random frame	procedural tests to satisfy a random search	TD	find a random search pattern
		BU	found a complex search pattern

figure 2.3

fully specified. Each of these 4 possible search types contains a group of tests that are specific to their own requirements.

Having found which search type is currently being used, an event sequence script is chosen and run to see if it matches the event sequence that is expected for the search type. A script in this case is a collection of commands that are to be executed by the currently active search type. They take into account some of the possible movements that should take place when a particular type of foraging sequence is being used.

Because the frame-slot decision mechanism is so simple, it is an easy matter to connect these frames together into groups that form a composition hierarchy of experiment types. This hierarchy is illustrated in figure 2.4. The **knowledge representation system frame** is the top most frame. Attached to it are other frames which are part of this frame. The **halves_experiment**, **quarters_experiment**, **random_experiment**, etc., can all share the same search types. Each of these search type frames has its own specialized test for a particular type of pattern, as shown in figure 2.3.

It is now clear from this collection of frames and slots that they can form networks that can be connected together. They all have the same essential structure and, therefore, can be changed or modified as the understanding of the bird's spatial patterning becomes clearer.

In the examples illustrated here the procedures were run when the slot was to be filled. When a particular frame or script has been selected as representing the current situation, the primary process in this type of reasoning system is simply filling in the details called for by the slots. This can be accomplished in several ways. Sometimes the type is directly inherited, but as in the **new_experiment** case there are several alternatives; for instance, the default experiment type can be used if there are no contradictions, or the attached procedure can be used to decide which experiment type to use. Default and inherited values are computationally inexpensive to use for filling in the slots; they do not require very powerful reasoning processes. This is really the power of the frame concept. However, should a frame or script prove inappropriate, attached procedures can trigger transfer of control to other frames(Minsky 1975). Another form of control transfer can be employed when a slot is found or changed. Trigger procedures implement event-driven, or data-driven, processing because they take over control only when certain events occur. For example,

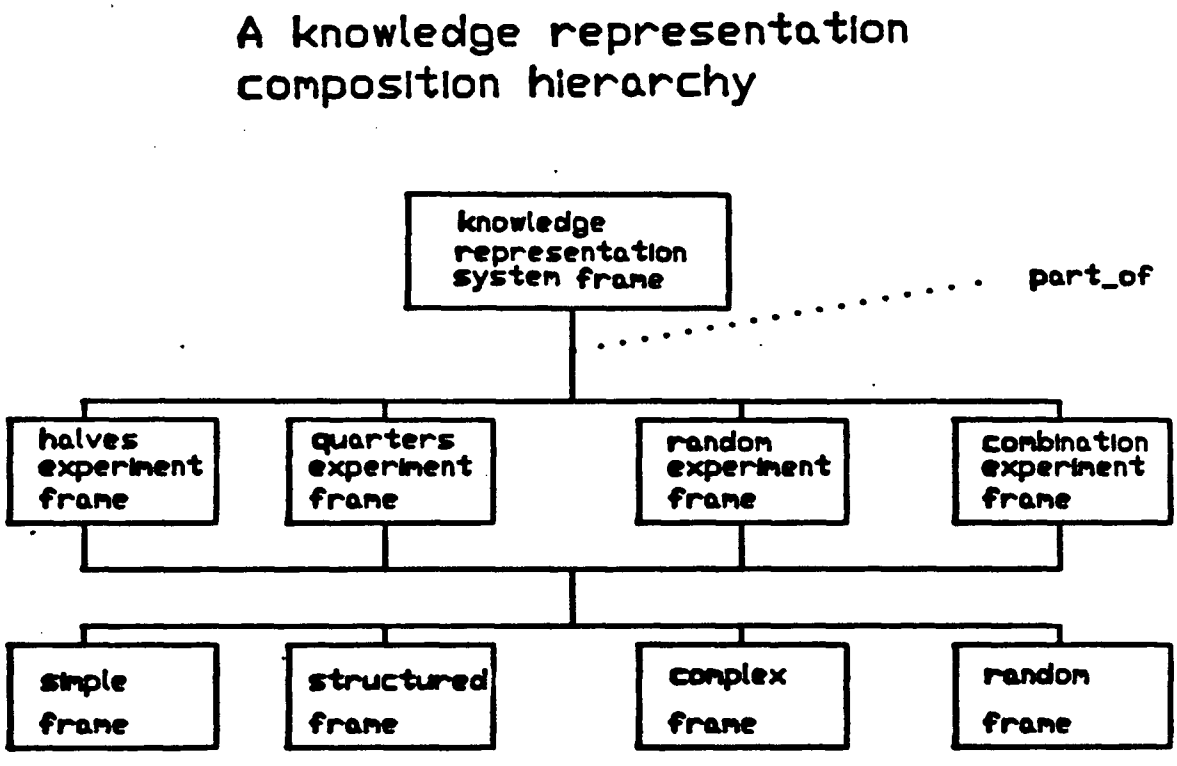

A knowledge representation composition hierarchy
knowledge representation system frame
part_of
halves experiment frame
quarters experiment frame
random experiment frame
combination experiment frame
simple frame
structured frame
complex frame
random frame

figure 2.4

the procedure in the default **simple** slot of the **new_experiment** frame is used to modify the list of alternative feeder configurations once a particular configuration has been chosen. In some situations these trigger procedures are used to decide what to do should the frame not match the current situation. A more complete review of frame based reasoning can be found in an article by Fikes et al, (1985).

2.8 Summary of Chapter 2.

- Frames, slots and scripts were introduced.

- A simple generic hierarchy was developed for a knowledge representation of patterns of hummingbird movements.

- Methodology was developed for rule definition and inheritance.

- The role of frames and scripts in describing heuristic knowledge was presented.

- Inverse compositional hierarchies were presented.

- The composition hierarchy for this knowledge representation system was presented.

Chapter 3

3.1 Combining a Knowledge Base and an Interpreter to Form an Expert System

The frame and script data structures form the basis of an expert system that is to be used to test the applicability of a set of coded spatial search patterns. Nearly all programming applications try to reach conclusions about a problem or set of problems. What differentiates an expert system from other application programs is the type of problem for which expert systems are appropriate. Problems addressed by traditional languages have all the necessary conclusions explicitly defined within the code; expert systems on the other hand, deal with problems that have a number of solutions and for which a programming algorithm would not be practical. Thus, the goal of an expert system is to generate the same solution to a problem as a human expert would when dealing with the same problem. An expert system must have four features in its structure (Walker, 1986). They are:

1. The capacity to represent a large volume of knowledge.

2. A non-procedural representation of knowledge (non-procedural means that the order in which information is presented to a program will not change the way it works).

3. The ability to add or remove information on a regular basis without affecting the structure of the program.

4. The ability to address problems involving inexact reasoning.

The order in which information is presented in traditional application programs often affects how that application performs. The types of problems addressed by an expert system involve many possible outcomes. Writing programs to fulfill all of these needs is an onerous task. Through the use of a knowledge base, an expert system provides a method of representing knowledge that is non-procedural in nature. The problems being addressed by the expert system change over time as new information is discovered. The expert system should be modular; that is, the structure of the program is not dependent on any one piece of information. The addition of new knowledge simply adds to the resources of the system, it does not require that the program be re-designed. Reasoning in an inexact manner is part of any discovery process, and an expert system must be able to assign weight, in the form of confidence limits, to any uncertain piece of information. Through the

assignments of weights it is possible to test one piece of information against another in an uncertain situation.

The expert system in this thesis is written in the programming language Prolog (Kowalski 1982). Prolog is different from C, Fortran or Basic. A Prolog program is a collection of rules and facts. Program solutions are arrived at by interpreting these rules and facts. One does not have to provide data typing or detailed instructions as to the flow of control; instead one defines possible solutions to a task, then provides the rules and facts that allow the program to select the correct solution. In other ways Prolog is the same as conventional languages. The purpose of a Prolog program is to accomplish a particular task. Prolog differs from conventional languages not in what it can do, but in how the correct solution is found.

The key to the reasoning process in an expert system is an inference engine. The inference engine uses information presented in the knowledge base to infer conclusions. The inference engine interprets the rules in the data base in conjunction with the control options in order to infer solutions to queries presented by the experimenter.

The use of interpreters for reasoning makes very powerful programming environments possible. Both Lisp and Prolog have this capability. The importance of the interpreter approach to an expert system design is its modularity. It is a simple task to add in features to an expert system by augmenting the interpreter, without affecting the knowledge base at all. Building an expert system now becomes a matter of customizing a program to fit the application. An individual can choose among the techniques that are desired and build them into the database; thus, the experimenter customizes the end product, rather than choosing among tools that do not exactly fit the task at hand. The following features were added to aid in customizing the expert system interpreter.

1. A general theorem prover that will work on any database of rules.

2. An uncertainty reasoner that incorporates uncertainties coded as probabilities into its reasoning mechanism.

3. An English language report generator that sets up a natural language interface to the reasoning predicates.

4. An instance lister that can find all instances of rules or rule fragments satisfying a general query.

The following is an example of the interpreters used in this thesis to keep track of a proof. This interpreter takes as inputs a set of queries and looks for a common solution for them all. While searching for a solution, **solve** keeps a log of its activities so that if a solution has been found, then its justification is present for the experimenter's examination.

This is how solve is coded in the Arity Prolog dialect:

1st predicate
```
solve((Goal1,Goal2),(Proof1,Proof2)) :-
!, solve(Goal1,Proof1),solve(Goal2,Proof2).
```

2nd predicate
```
solve(Goal,implies(Proof,Goal,Body)) :-
clause(Goal,Body),solve(Body,Proof).
```

3rd predicate
```
solve(true,true) :- !.
```

4th predicate
```
solve(Goal,implies(system,Goal)) :-
functor(Goal,Name,Arity),
system( Name / Arity ),
call(Goal),!.
```

The first predicate **solve** takes in a sequence of goals separated by a "," operator and outputs a sequence of proofs. It does this by first solving **Goal1** and obtaining **Proof1**. It then proceeds on to **Goal2** and returns **Proof2**. All of the structures built up while finding **Goal1** are passed on to **Goal2**. If the second call to **Goal2** fails, backtracking is automatic, and the first call to **Goal1** is re-invoked.

The second predicate solves a goal by finding a clause body in the data base and then finding a proof of that body. The head of a clause is all of the information to the left of the ":-" and the body is everything to the right, up to the ".". The whole thing together makes up a predicate. **Implies** is simply a place holder to record a particular call to **solve**. There is no "!" or cut(a cut stops backtracking) because one must be able to find alternate clauses in which **Goal** exists if the original call to **solve** fails.

The third predicate takes care of facts. All facts in a Prolog data base are considered to be rules. The way facts are differentiated from rules is by making them all true. This predicate does just that.

The fourth clause proves system predicates, e.g., $5 < 6$. **Functor** simply retrieves the name and number of arguments of a system predicate.

Another type of **solve** can be constructed to perform a completely different task. This **solve** does not return a proof but rather a conditional probability. That is, it finds the minimum of some function's certainty factor(cf). Let this function be $f(A, B)$. This **solve** will try to find

$$cf(\ A\ \wedge\ B) = min(cf(A), cf(B))$$

The new **solve** is included here to demonstrate that the basic **solve** interpreter can be quickly modified to look at an entirely different type of data base, in an entirely different manner. This means that the inferencing is performed on a data base that now contains rules, facts, and uncertainties. The interpreter looks like

```
solve(true,1).
solve(Goal1,Goal2,C) :-
solve(Goal,C1),solve(Goal2,C2),ifthenelse((C1<C2),(C=C1),(C=C2)).
solve(Goal1,C) :-
clause(Goal,Body,C1),solve(Body,C2),(C=C1*C2).
```

Here the predicate **solve(Goal,Certainty)** is true when **Goal** is satisfied with a certainty factor called **Certainty**. The interpreter here computes the combination of certainty factors(as shown above) as the minimum of the certainty factors of two goals. This is an arbitrary choice. Since this interpreter can be modified in a direct manner, there are many other ways one can reconfigure this inference engine so that it may perform its reasoning.

The statements about facts concerning the types of search patterns to be tested are entered as lists of arguments. For example, if one wished to insert into a Prolog data base - information about what constitutes a bottom quadrant search strategy - one would proceed in the following manner.

```
search_strat(Sample,btm_quad):-
        search_type(Sample,complex),
        search_bound(Sample,low_tri).
search_type(Sample,complex):-
        search_range(Sample,high),
        search_success(Sample,HITS),
        HITS > 7.
```

The above example would be one of many search strategy types included in the data base. Its statements form a simple tree-like structure that will be searched in a specific manner. To make the tree's structure clearer abbreviations will be used. These are: SS = the set of all search strategies.

SSBQ = search strategy in the bottom quadrant.

ST = search type.

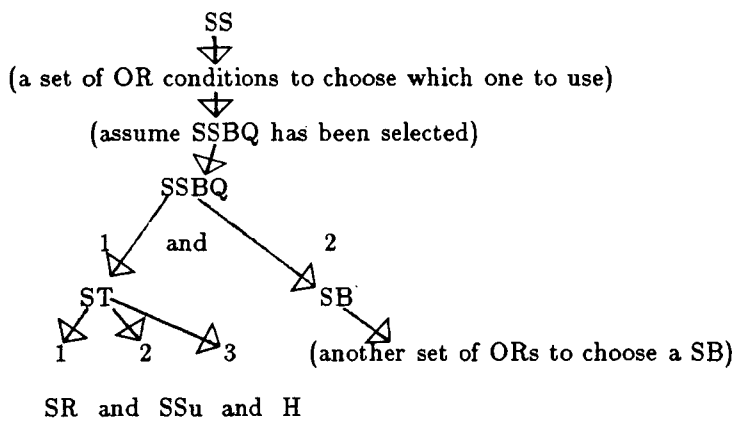
SB = search bound.

SR = search range.

SSu = search success.

H = hits > 7.

The tree that is traversed to satisfy this goal looks like the following:



The logic that is used to satisfy the goal SSBQ uses a collection of logical "ands" that must be satisfied in a particular order. The first thing that SSBQ must find to be true is that ST "and" SB are true. This is carried out by moving left to right across the tree. The "1"s and "2"s indicate the order of testing, at a particular level in the tree traversal, in which things must be found to be true. In this example, in order for ST to be true SR and SSu and H must be true. If this level has succeeded then ST and SB must be found to be true. If everything has been found to be true at this level and the goal SSBQ has been found to be true, then the answer "yes" is returned to the query. The answer "yes" is not very useful for uncovering the reasoning that has just taken place; another procedure is necessary to interpret the result by providing the facts and rules that were used along the path towards the goal. The query **how** has been coded to provide the reasoning to a proof in a form that can be easily interpreted by the experimenter. **How** looks like:

```
how(Goal) :- (solve(Goal1,Goal2,Proof1,Proof2),nl
interpret(Proof1,Proof2);
writeln('... THIS IS FALSE'),fail),!.
```

How uses **solve** to find the goals and then calls **interpret** to present the reasoning in a readable form.

It is now possible to enter a query about the type of search strategy being used, and obtain all of the reasoning involved. At this point in the execution, all of the tree structure has been found to be true in the data base, the proof interpreter has collected the reasoning, with suitable English statements, and is ready to print them out on the console.

The response to **how(btm_quad).** is as follows:

search_strat(sample1,btm_quad)

CAN BE SHOWN USING THE RULE ...

search_strat(sample1,btm_quad)

< – – – this is implied from

search_type(sample1,complex) , search_bound(sample1,low_tri)

FURTHERMORE ...

search_type(sample1,complex)

CAN BE SHOWN USING THE RULE ...

search_type(sample1,complex)

< – – –this is implied from

search_range(sample1,high),search_success(sample1,8),8 > 7

FURTHERMORE ...

search_range(sample1,high)

IS A FACT FROM THE DATABASE

search_success(sample1,8)

IS A FACT FROM THE DATABASE

8 > 7

IS A BUILT-IN FACT

search_bound(sample1,low_tri)

IS A FACT FROM THE DATABASE

The format of the response shows how the intepreter works backward from a given goal to the smallest atomic structure(the data base facts) while filling in the reasoning process. It should be noted that the smallest atom of information need not be static facts. They may be other goals that are needed to be tried in a different goal context. This is how a dynamic reasoning structure can be built into the expert system.

Once again the modularity of the interpreter makes this type of backtracking simple. As discussed in Chapter 2 this relation forms an inverse hierarchy that allows goal oriented forward reasoning and backward reasoning from the instantiation of the goal **how** .

Finally, one would like to see how many other items in the data base can possibly satisfy a general query about **search_strat**. This is particularly useful when there are large numbers of candidates for solving a particular problem. To do this, another procedure was written to provide the information. It is called **follows_from**. The query **follows_from(search_strat(X,Y),Z)** will search all data base samples **X** finding all search strategies **Y** that can be found in the body of rules **Z** that are contained in the expert system's data base. The response to this query is as follows:

THE PROPOSED RULE:

search_strat(X,Y)

< - - -is implied from

Z

HAS THE FOLLOWING INSTANCES OF RULES IN THE DATABASE:

search_strat(X,diagonal)

< - - -is implied from

search_band(X,less_than_2)

search_strat(X,top_half)

< - - -is implied from

search_type(X,simple) , search_bound(X,top_half)

search_strat(X,btm_half)

< - - -is implied from

search_type(X,simple) , search_bound(X,btm_half)

search_strat(X,btm_quad)

< - - -is implied from

search_type(X,complex) , search_bound(X,low_tri)

search_strat(X,top_quad)

< - - -is implied from

search_type(X,complex) , search_bound(X,up_tri)

THERE ARE NO MORE INSTANCES

The reasoning processes are now all in place in the form of small interpreters that perform a variety of tasks. The search types in the data base have the frame like structure outlined in chapter 2, and can be quickly traversed by the Prolog reasoning mechanisms. The command scripts are the last piece to include as part of the reasoning system.

3.2 Including Scripts in the Knowledge Base of the Expert System

Having demonstrated that a reasoning mechanism can be built from simple interpreters, the next task is to incorporate the frame-script ideas developed in Chapter 2. As has been previously stated, the simplest level of data structure or "literal" as it is referred to in Prolog, is the fact. Facts do not have to be static statements but can be triggers to other procedures. These coded procedures are the frames and scripts. Thus, when the reasoning mechanism is trying to verify a goal, part of that goal can be a set of procedures that are run in some pre-determined sequence according to a script. These procedures would look at the available data using an appropriate numerical technique. The types of numerical techniques employed are outlined in detail in Appendix 1.

Another different type of script that is employed to track a foraging sequence is that of a graphics interpreter. This script is run through the Computer Aided Design(CAD) package called Autocad. When this graphical interpreter is invoked any application can be run as a script. The command to run any predetermined script is called **SCRIPT** . When this command is issued the file containing the script is asked for and the script steps are executed. A simple script to draw a 3D orthographic projection of a foraging trajectory is shown as follows:

LAYER NEW 1 COLOR YELLOW 1 SET 1

ELEV 0.00 0.1

LINE 1,3

1,4 2,4 2,3 3,3 3,4 3,5 4,5

ELEV 0.10 0.1

LINE 1,4

1,2 1,3 1,4 2,4 2,3 3,3 3,2

The LAYER command sets up a new plotting layer in the screen. This layer has various attributes that can be set: such as its name, which is the number 1, its COLOR, which is YELLOW, its height from some predetermined baseline, and its thickness. Thickness here is 0.1 and is the z axis height. LINE draws a line from 1,3 to 1,4 etc.. This is the simplest script possible under this interpreter. Other far more complicated scripts that invoke the running of other programs, such as the warping program, can be incorporated into this type of script.

The reasoning procedures running under Prolog can be made to incorporate other scripts with their own reasoning and data investigation. This graphics interpreter is simply one example. The hierarchy of these data structures forms a network of frames like those discussed in chapter 2. The level control for all of these frames and scripts is neatly handled by the simple interpreters demonstrated earlier.

3.3 Coupling the Expert System with the Numerical Software

Systems that link together symbolic and numeric processing into a single composite function are called coupled systems and these in turn are called referenced systems if the numeric processes are specialized and the knowledge based system reasons about the applications and results. Coupling of these two separate forms of computing has become one of the emergent areas in AI (Kowalik, 1986).

The knowledge representation system developed to study this foraging environment moves the investigating procedure another level beyond coupling and referencing. Most coupled systems that employ a combination of symbolic and numeric analysis have a fixed structure within which decisions are to be made by the investigator concerning the appropriateness of a subset of the pre-existing available models for a particular application.

The system presented in this thesis seeks to find the appropriate structure that will suggest which class of models should be used in a particular situation. This goal was accomplished mainly through the use of animation which permitted different types of searches to be classified into either simple searches or complex searches. These were broken down further into diagonal, quadrant, and other searches. The classified searches were then coded and tested against the actual database. Cutting and fitting of enumerated search patterns permitted incremental discovery of the symbolic and numeric components that are necessary for designing a spatial foraging model.

There is a continual tradeoff between formal analysis of the data and qualitative reasoning. The experimenter must assess the foraging behaviour as it changes in response to the learning process. This requires constant switching between these two modes of analysis. This problem was summarized by Chandrasekaran, 1983. He states that

"Even if we have a complete mathematical mode of a situation, this by itself may be insufficient for many tasks, since qualitative reasoning is required to interpret the numerical values of the various problem variables. Thus the solution of complex problems often compels us to switch back and forth between formal analysis and qualitative reasoning."

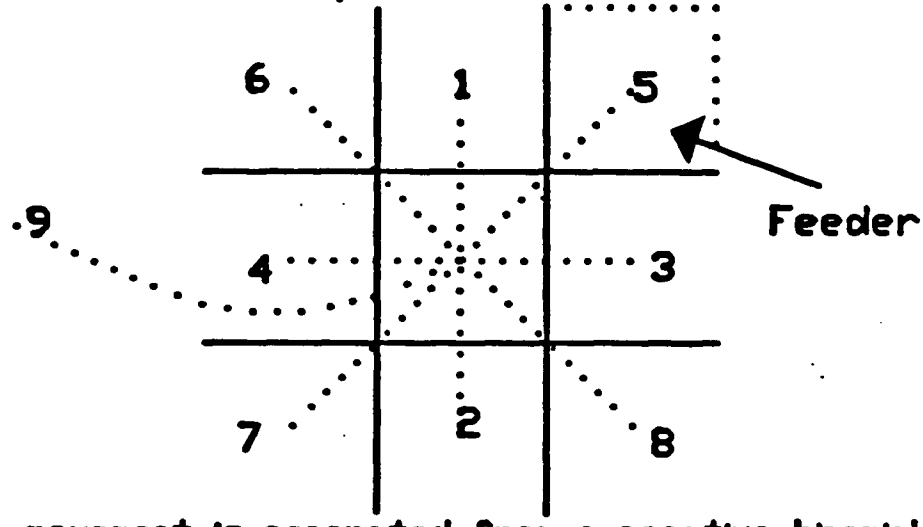
There are two numerical problems to be resolved in order to properly analyze the data.

3.4 Modelling Irregularly Sampled Spatial Sequences in One Dimension

In the first numerical situation one must be able to deal adequately with a foraging trajectory that follows an irregular path. The birds traverse the grid in a great variety of ways. This makes it very difficult to compare different trajectories used by different birds, or trajectories for the same bird at different times. In addition, one wants to be able to construct spatial strategies as tests, and compare them to those used by the birds, to see if the bird's behaviour is consistent with the proposed strategy. Towards this end a simple spatial movement model was constructed to imitate a hummingbird's foraging movements. The model was started at the location where the bird began the trial and was alotted the same number of moves that the bird used during the targeted trial. A set of model movements was generated using a negative binomial probability of an individual move. The f failures before s successes of finding a filled feeder were based on the expected values from observing the bird during the previous trial. Variations on this assignment consisted of using the expected f and s values for n previous trials, or selecting the p most successful trials in a group. Figure 3.0 illustrates the 9 different types of moves possible from a current location.

The move direction is an integer, and relies on integer arithmetic to truncate the floating point probability, multiplied by 9 to allocate a move. The 9 move is a movement to some other starting point on the grid. Each move represents the movement from one feeder to another. This generates a foraging trajectory through a trial by a foraging model. One now wishes to compare the model trajectory to the bird's trajectory to see if they match in some systematic way.

An illustration of possible model movements



Each movement is generated from a negative binomial distribution having f failures before s successes at locating a filled feeder

move direction = 1+9*(prob of move)

figure 3.0

3.5 Partitioning a Foraging Pattern while Conserving Adjacencies

Before developing the appropriate mathematics to match these two trajectories, a second part of the irregular sampling problem must be resolved. Mapping a spatial trajectory into a location for each of the moves in a trial serves to illustrate the foraging trajectory pathway. However, the use of this straight line mapping destroys the location adjacencies. This means that the spatial location has, in the first case, simply become a place mark along a one-dimensional axis. Using this methodology, one can see if the foraging trajectories match, or whether two foraging strategies match, but conclude little about the specific spatial partitions that the bird used. One of the most important questions raised by this experiment is the manner in which the bird partitioned up the grid in order to learn its structure. In order to see and analyze this partitioning, the grid must be considered to be a map where adjacencies are preserved.

The bird concentrates its search in sections of the grid where it expects filled feeders to be located and as a result the grid is unevenly sampled. The amplitude measure at each spatial location in this situation will be the total number of times per trial spent at a location. Thus, a temporal sampling strategy has been integrated over time to permit a look at the spatial distribution of searching efforts. If the experimenter has a hypothesis about how the bird's efforts will be distributed spatially, then the situation becomes a question of 'How does one go about testing this hypothesis against the recorded behaviour?' Figure 1.0 represents a 'checkerboard' design composed of 4 filled feeders in each of eight groups which are set up as shown. The filled feeders all contain the same amount of sugar water and are filled only at the beginning of a 40 s trial. To test some aspect of the bird's spatial partitioning, one would select a grid that has been sampled over a number of trials, 10 in this case. From this map one computes the total number of times the bird visited each feeder. The question that arises now is 'How does one compare an equally sampled spatial grid to an irregularly sampled one?' The answer to this question is that somehow one must 'regularize' both to a common measure and then compare the differences.

3.6 Warping Irregularly Sampled Sequences in One Dimension

The methodology chosen to compare the one-dimensional trajectories is called warping. Warping means the mapping of one trajectory onto another by shifting the points along the location number axis. The shifting algorithm tries to improve the degree of correlation between trajectories. The amount of shifting is penalized using a cost function. A high degree of correlation produces a low cost, and a low correlation a high cost. In addition to the cost function there is a penalty function which is proportional to the square of the shifting distance. This means that the further one must shift the model to fit the behaviour, the more one is penalized. Adjusting the effect of the penalty is a scaling parameter. This feature allows one to have control over the amount of warping that is considered by the experimenter to be permissible. To find the optimal amount of warping dynamically, the dynamic programming algorithm developed by Bellman(1957) was used. The combination of these two algorithms was part of a thesis written by Kim Roberts(1984) in the department of electrical engineering at U.B.C.. The computer program that has been implemented was listed in the back of this thesis and written in a dialect of Pascal. A complete discussion of the structure of the algorithm as well as the cost and penalty functions are given in Appendix 1.1.

To illustrate how this warping works a sample of two behaviour trajectories from separate days were warped onto one another. The experimental design in figure 1.0 has its feeders numbered from 1 to 64 beginning at the top left hand corner and proceeding across the row and then down to the next row. The numbering is shown in figure 3.1.

In this test, the two trajectories are required to have the same number of elements. These are numbered from 1 to T. The amplitudes of each sequence are equal to the total number of visitations by bird 1 in each of the 16 sectors for the first 10 trials on June 27 and 28 respectively. A sector is a grouping of 4 feeders. The first sector is made up of feeders 1,2,9 and 10 and the second is made up of 3,4,11 and 12 and continues across each row. What one wishes to discover is whether it is possible to move the positions of the first trajectory, labelled **a** to fit the second trajectory, labelled **b** . The results of these efforts are presented as a graph in figure 3.2. The x and y axes have units that run from 1 to T. It should be noted here that T can be either 16, or 64 depending on whether one is using sectors as the spatial division, or individual feeders. In this first case T is 16, denoting the number of sectors on the grid. If the elements of the first trajectory are denoted

An illustration of the feeder numbering

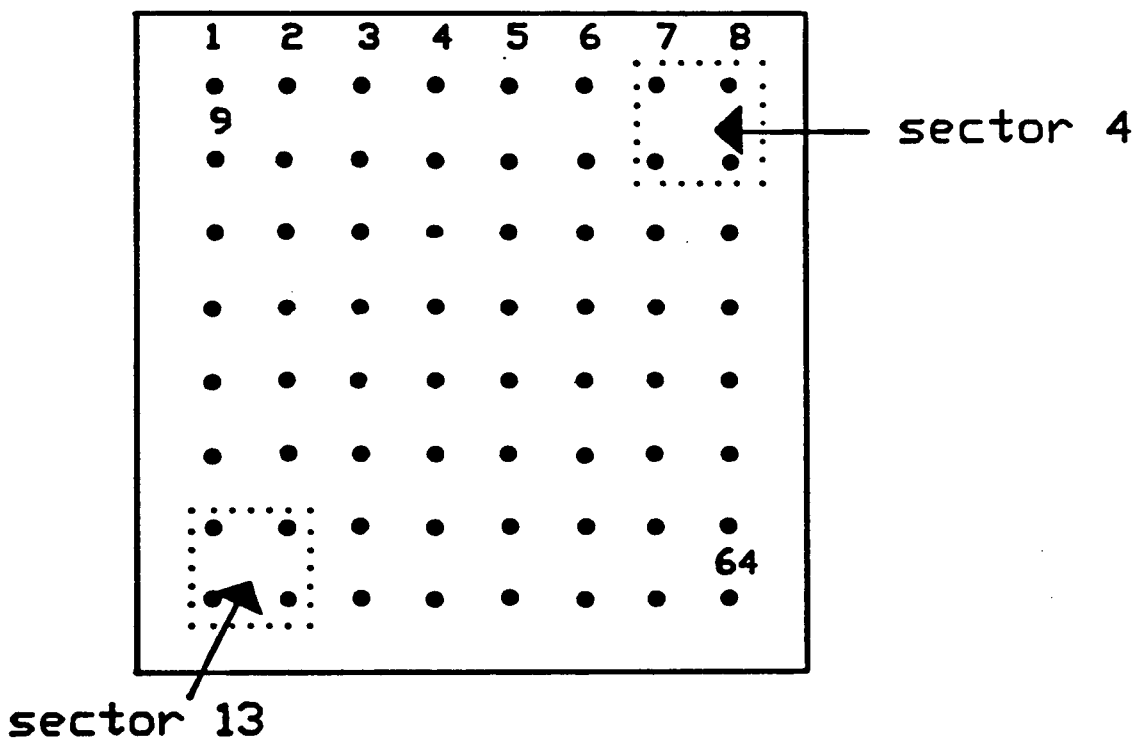


figure 3.1

An illustration of a one-dimensional
warping function and the adjustment window

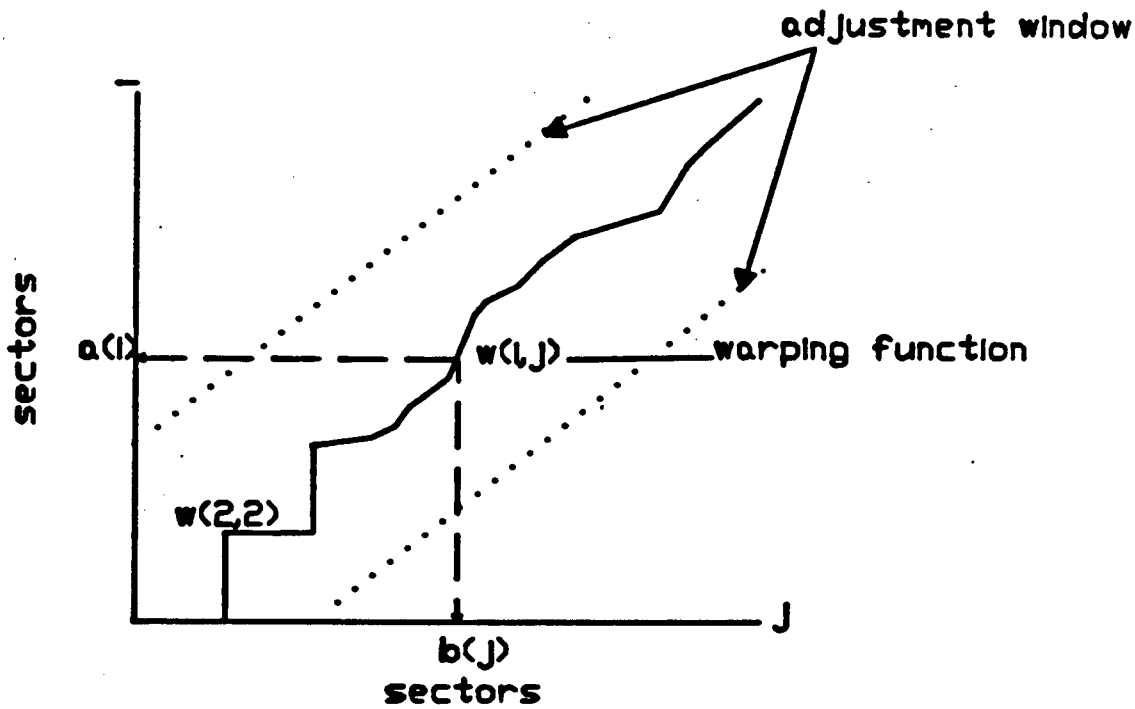


figure 3.2

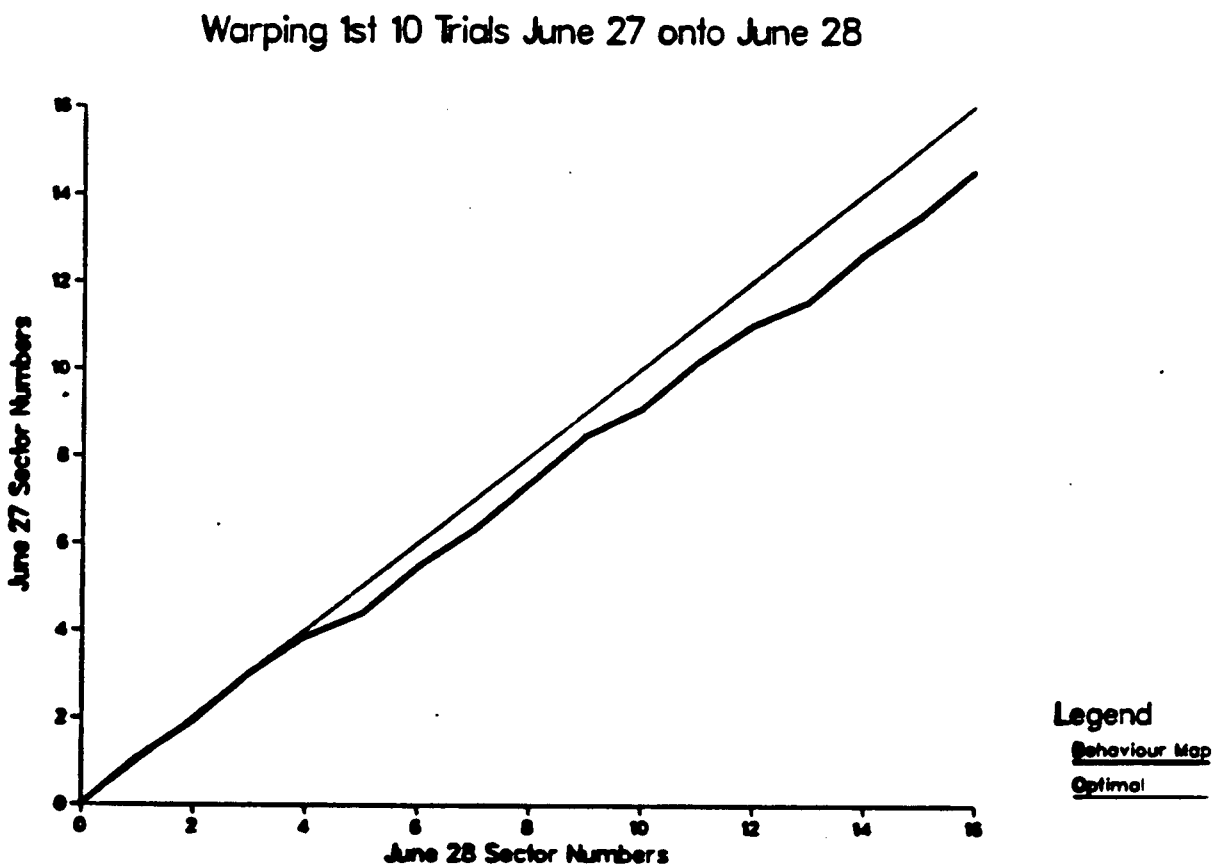

Warping 1st 10 Trials June 27 onto June 28
June 27 Sector Numbers
June 28 Sector Numbers
0
2
4
6
8
10
12
14
16
Legend
Behaviour Map
Optimal

figure 3.3

as $a(i), i = 1,...,T$, and those of the second as $b(j), j = 1,...,T$, then the warping function $w(i,j)$ is such that $a(w(i,j))=b(j)$. It can be seen that $w(i,j)$ is an index, indicating which element of **a** corresponds most closely, in terms of amplitude with **b** , i.e., $a(i) = w(i,j)$. If the elements of **a** corresponded exactly with those of **b** then there is no permuation of indices. This would make the graph of $w(i,j)$ a straight line angled at 45 degrees from the x axis. The graph of **w** has kinks in it corresponding to the differences between the two sequences to indicate where the indices changed. These results are indicated in figure 3.3. One can see from the graph that the two sequences are similar, but differ in a gradient-like manner as one moves down the grid. What this means is that there was more specialization on June 28 in the lower segments of the feeder grid.

3.7 Warping Irregularly Sampled Sequences in Two Dimensions

The two dimensional warping that is used to regularize the spatial sampling grids is motivated in a different manner from the one-dimensional case. One begins with the underlying sampling assumptions. The grids used in these experiments are homogeneous in their designs, but non-uniform in the way they were sampled. The bird favoured some sections of the design over others. This non-uniformity is typical of most applications. The sampled points on this grid are assumed to arise as the result of a perturbation of some regular sample set. The 'regular sample set' chosen in this case is the hexagonal lattice. Hexagonal lattices have some unique properties that make them very useful as a basis, and these properties are discussed at length in Mersereau(1979). The computer algorithm that is used to implement this lattice mapping was developed by James Clark(1985) as part of his PhD thesis in the department of electrical engineering at U.B.C.. In his algorithm Clark assumes that the perturbations that arise in the data set are small enough that the perturbed point will remain somewhere within a 60 degree sector about its original position. A realization of this type of lattice and its perturbation can be seen in figure 1.2(a) in appendix 1.2.

Clark's algorithm begins by assuming this hexagonal basis, then searching for its center, denoted as H0. One now labels this perturbed point as x, and searches for x in the experimental sequence at the closest point for which reconstruction can be obtained. When this point is found one must then map it onto H0. This is repeated six times, once for each sector line in the hexagon. This means that once the point xT has been found which maps onto H0, one must then find the other 6 points

of the hexagon . Having found this first hexagon in the center of the lattice, the search then spirals outward following the pattern of figure 1.2(b) in appendix 1.2. The outward spiral stops when the edge of the grid is encountered.

Figure 1.2(c) is a raw data sequence composed of ten foraging trials performed on June 27, 1985. The amplitudes are the total number of visitations to a feeder for the entire ten trials. Figure 1.2(d) illustrates the 'regularizing' effect that the two-dimensional warping applies to the grid. The diagonal feature and the bottom right quadrant are easily recognized as the principle places of foraging effort. This makes it a very simple matter to decide which of the foraging stereotypes are currently being employed in the seach strategy of the hummingbird.

This simplification of the grid's patterns allow the pattern matching procedures, such as **bottom_quad**, to be found much more accurately. When the search procedures are given the original grid to work on, the answers returned indicated that everything in the data base was at least partially satisfied. When the grids were warped, the search movements were more concentrated and greatly simplified; consequently, the search type differentiation became much more accurate.

3.8 Searching for Rules in a Database

When the foraging sequence rules are searched for in an actual recorded foraging sequence, one must be able to find the largest group of consecutive movements that will fit into a rule. For example, a **diagonal** can be any sequence where row and column sums are equal. Typically, a bird moves back and forth inside this bound as it searches the grid. This means that the trajectory may qualify as a **diagonal** trajectory if only one transition occurs, say from 1 to 10 in figure 3.1. Stated in mathematical terminology, one wishes to find the maximum subset of points(the rule) in a complete larger set of points(the foraging trajectory). This problem was solved in 1928 by the mathematician E. Sperner and proposed as a lemma. Mathematically the problem can be stated as follows:

Let F be a family of distinct subsets of $\{1,2,...,n\}$ such that

$$S, T \in F \Rightarrow S \not\subset T. \qquad 3.1$$

The question to be answered is; how large can $|F|$ be?

Sperner proved that

$$|F| \leq \binom{n}{\lfloor n/2 \rfloor}. \qquad 3.2$$

Lubell(1966) provided a very simple proof to this problem. He developed it as follows:

Let A denote the family of all 2^n subsets of $\{1,2,...,n\}$; let a family be called F feasible if it satisfies 3.1. With each family, F, feasible or not, associate the vector $(x_S : S \in A)$ defined by

$$x_S = \begin{pmatrix} 1\,, & \text{if } S \in F, \\ 0\,, & \text{if } S \notin F. \end{pmatrix}.$$

Therefore,

$$|F| = \sum_{S \in A} x_S\,. \qquad 3.3$$

A family of sets $T_0, T_1, ..., T_n$ with

$$\emptyset = T_0 \subset T_1 \subset ... \subset T_n = \{1,2,...,n\}$$

is called a chain. Since there are $n!$ distinct chains, each $S \in A$ is included in $|S|!(n-|S|)!$ of them. Furthermore, F is feasible if and only if

$$\sum_{S \in C} x_S \leq 1 \text{ for every chain } C. \qquad 3.4$$

The sum of all these $n!$ inequalities 3.4 appears as

$$\sum_{S \in A} |S|!(n-|S|)!x_S \leq n!$$

or, equivalently,

$$\sum_{S \in A} \frac{1}{\binom{n}{|S|}} x_S \leq 1.$$

Since every x_S is nonnegative and every

$$\binom{n}{|S|} \leq \binom{n}{\lfloor n/2 \rfloor},$$

one has therefore,

$$\sum_{S \in A} \frac{1}{\binom{n}{\lfloor n/2 \rfloor}} x_S \leq \sum_{S \in A} \frac{1}{\binom{n}{|S|}} x_S.$$

This means that 3.4 becomes

$$\sum_{S \in A} x_S \leq \binom{n}{\lfloor n/2 \rfloor}$$

Looking at 3.1 it can be seen that this is the desired result.

The requirement that the sequences be contiguous greatly restricts the subspace that must be searched to obtain a maximal set satisfying one of the assigned rules. Indeed, the longest maximal sequence is restricted to the largest dimension of the grid which is 8. Thus, when recursively searching a list of a maximum of 45 movements the computational slow-down associated with combinatoric width and depth searches can be avoided by applying this dimensional constraint in the search.

3.9 Summary of Chapter 3

- The motivation and methodology for combining knowledge and a interpreter to form an expert system was presented.

- Important features of the implementation of the software for this application were presented.

- Examples of the theorem proving, using the Prolog dialect were presented.

- A demonstration of how scripts are included in this expert system was presented.

- Numerical algorithms for matching foraging trajectories to one another were developed for one and two dimensions.

- The theorem for finding maximal subsets of rule fragments was developed and a sample of subset bounds was presented.

Chapter 4

4.1 Implementing the Knowledge Representation System and Pattern-Matching Algorithms on a Micro-Computer

The knowledge representation system was implemented on an IBM AT micro-computer that used MS-DOS 3.1 as its operating system. The chief drawback of this system is that it is not concurrent, that is, only one task can run at any given time. The Xenix operating environment is concurrent but does not support applications running under MS-DOS. In addition it is not possible to implement software designed in a Xenix environment in a MS-DOS environment. Various attempts were made to implement a concurrent environment for this knowledge representation system with little success. In terms of data aquisition this limitation is the chief bottleneck in the system design. Once the data collection has been made, the MS-DOS environment works exceptionally well at controlling the software and hardware used by the system.

4.2 System hardware design

This system was implemented on an IBM AT. This micro-computer had two 20 megabyte(mb) hard disks with 1.152 mb of read access memory (RAM). The RAM consisted of 640 kb of main memory and 512 kb of additional memory on a Tecmar board. An 80287 co-processor was used for all floating point operations. The access time on these disks for a single read/write instruction was 32 milliseconds(ms). The system clock ran at 9 megahertz(mh). The system was also equipped with an Enhanced Graphics Adapter with 256 kb additional memory. The monitor used was a Sysdyne color graphics monitor having 640×350 pixels supporting 16 colors and 4 screens. There were two serial ports and one parallel port active at any one time. The first serial port was connected to a local network having campus wide access while the second serial port was connected to a Hewlett Packard 7475a six pen plotter. The parallel port was connected to a dot matrix printer. The speed of this AT system made it an ideal development system for the numerically intensive operations as well as the symbolic searching operations. However, as the database of pattern matching procedures increased, the system slowed considerably, indicating that the implementation of a complete pattern

matching system would require additional stand alone pcs to aid in distributing the computational load.

4.3 System control design

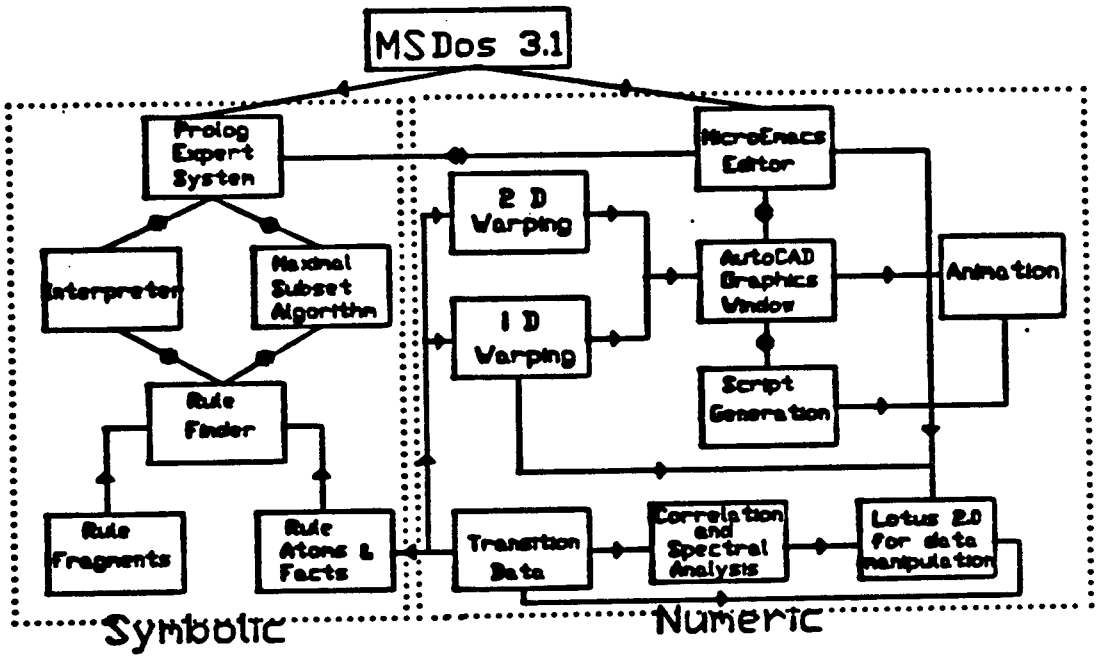
Figure 4.0 illustrates the computing system's design and hierarchical structure. The system editor is an implementation of the Unix EMACS editor, called MicroEMACS. This editor allows graphics applications, and other applications to be run from a command line in the text editor. When the graphics application is completed, control is passed back to the editor with the correct attribute bytes. This restores the screen to its original text orientation. This feature is also of considerable advantage during system development where toggling in and out of the editor is time consuming and unnecessary. Since this editor is written in the C language, and the source code is available, customization of its performance for any system design is a simple process.

4.4 Using Graphics Windows to Run Scripts

The principal graphics window for this system is the AutoCAD software package. It can run in two modes, interpretive or batch. The batch mode, discussed in Chapter 3, permits previously designed applications to be saved as a script of instructions in a file. These instructions can be developed as either a two-dimensional plot or as a three-dimensional plot. Scripts can contain shell command instructions to the DOS operating system that invoke other software. This means that several types of numerical analyses can be run in a predefined manner and their graphical output displayed as the results unfold. Through a **shell** command it is possible to return to EMACS, change these scripts, and then rerun them to observe changes in the results.

Two other commands were found to be useful for controlling the rate at which the scripts were executed. The **delay** command slows down the script execution so that the details of the drawing can be seen. This was of particular use when viewing the animations of the foraging sequences, because the details of the individual movements were lost when drawn at the system speed. When a delay of 10 ms was applied, the details were more easily followed by the eye. The second script command is **resume**. If a **Ctrl-C** is typed in from the keyboard, the script suspends execution. If the **resume** command is entered, then the script continues where it left off. This was particularly

An illustration of the software flow of control



▶ = direction of information flow

figure 4.0

useful for obtaining partial plots of results. The system also uses the concept of layers of graphical overlays. For example, it is possible to run a script that conforms to one set of commands, or frame specifications, and define it as a named layer. If one issues a **freeze** command and names the current layer, it becomes invisible. If one labels another layer as active and reruns the script with its modifications it will re-plot the results. When this script has finished, the first layer can be unfrozen and the two viewed together and the differences visually compared. Many layers can be thawed and frozen as the analysis proceeds. There is another option that allows creation of custom menus. This option is especially useful if a variety of scripts are to be tried on a particular data base. It is a simple process to capture data as they are collected and animate them on the screen. Animation proved to be a very useful tool for picking up slight variations in behaviours as they evolved. Being able to animate the trajectories taken by these birds as they learned the experimental design is of premier importance for understanding what has actually occurred and what aspects of the behaviours are pivotal to the knowledge representation system.

4.5 Using scripts in a graphics window to animate behaviour patterns

Figure 4.1 illustrates how the animation script works. Each box of 4 angle brackets represents the four filled feeders in a sector. At the mark labelled t0 the feeder array itself is drawn before any trials are animated. Time progresses upward from here to t1 and onward following the direction of the arrow. Each movement by the bird is represented by a single band segment drawn from where the transition started, to where it ended. The bands are all drawn at the same level during a trial. Each subsequent trial is displaced upward at a constant distance, beginning at the base of the feeder array t0, and ending at an absolute displacement equal to the number of trials animated. As the data are drawn by the script it appears as though the bird is pulling a segmented ribbon behind it.

Illustrations of how the data were animated to view the June 26, 27, and 28 1985 foraging sequences for the first bird is presented in figures 4.2(a to l). A complete trial consisted of up to 45 connected segments and represents the entire behaviour sequence. The absolute displacement in each figure is 10, indicating that each animation contains 10 trials. Figure 4.2(a through d) represents trials 1 to 10; 11-20; 21-30; 31-40 respectively from June 26 for bird number 1. The learning by the birds as time progresses is clearly illustrated. In the first figure(4.2 a), the bird is seen to use

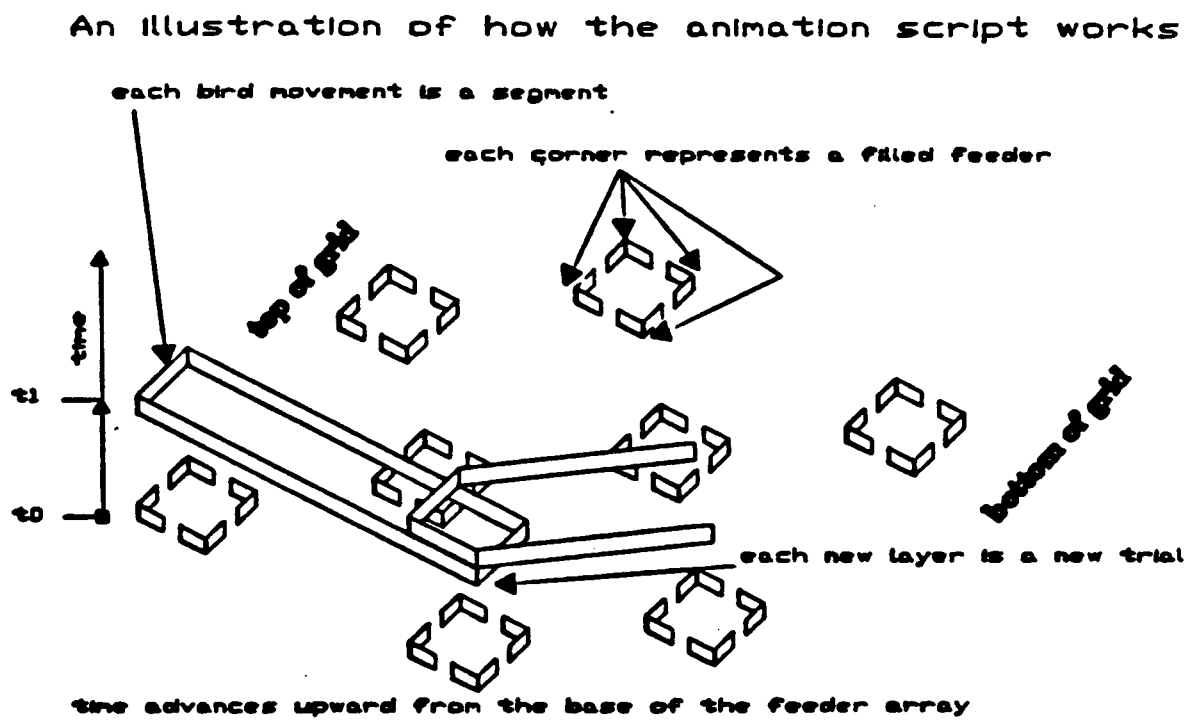

An illustration of how the animation script works
each bird movement is a segment
each corner represents a filled feeder
top of grid
time
t1
t0
bottom of grid
each new layer is a new trial
time advances upward from the base of the feeder array

figure 4.1

Animation of the foraging path
for bird 1, June 26, 27, 28
blocked into 10 trial increments

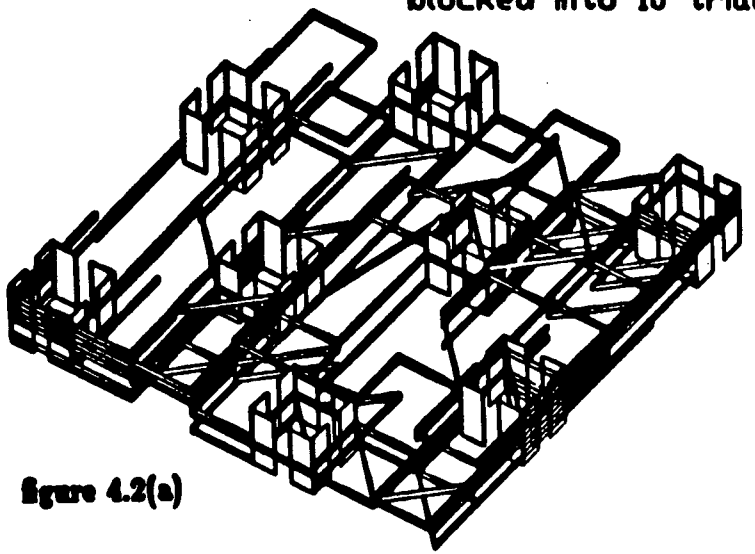


figure 4.2(a)

trials 1-10

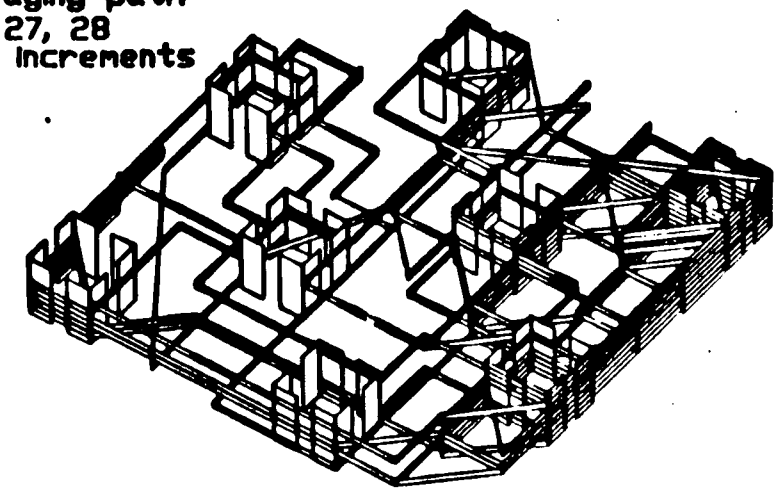


figure 4.2(c) trials 21-30

June 26

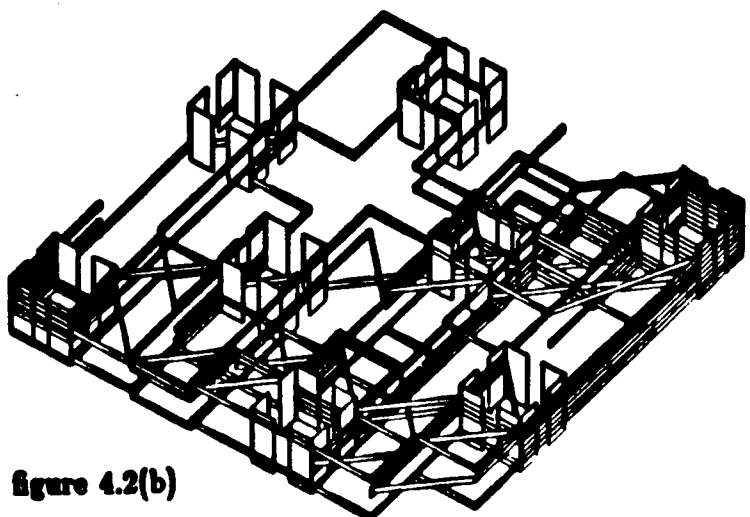


figure 4.2(b)

trials 11-20

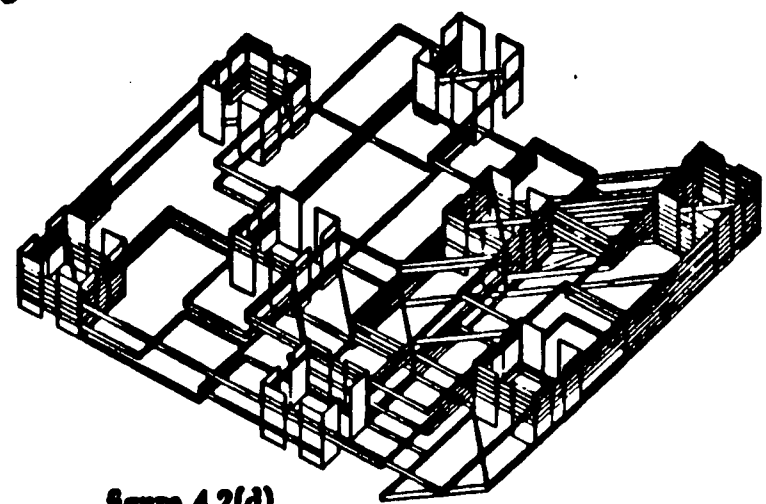


figure 4.2(d)

trials 31-40

figure 4.2(a,b,c,d)

Animation of the foraging path
for bird 1, June 26, 27, 28
blocked into 10 trial increments

figure 4.2(e)

figure 4.2(g)

June 27

figure 4.2(f)

figure 4.2(h)

figure 4.2(e,f,g,h)

Animation of the foraging path
for bird 1, June 26, 27, 28
blocked into 10 trial increments

trials 1-10

figure 4.2(i)

trials 21-30

figure 4.2(k)

June 28

Note: the filled sectors
have been shifted
1 sector to the right
for this set of trials

trials 11-20

figure 4.2(j)

trials 31-40

figure 4.2(l)

figure 4.2(i,j,k,l)

a simple pattern, across a row and down a column with revisitation most notably to the bottom of the feeder grid. In the second figure(4.2 b), the bottom two rows are featured search areas, while the top right quadrant is neglected. In the third figure(4.2 c), the across and down search strategy has been replaced with a diagonal strategy moving from middle right to bottom left sectors as well as an indication that the principle diagonal components near the bottom have have been discovered. The fourth figure(4.2 d), is much the same as the third with an expansion of search to include the middle part of the grid.

The second day, June 27, figure 4.2(e through h), began with the bird returning to the experiment after having been left in the experiment room with a single unlimited supply feeder for 17.3 hours. What is suprising here is that none of the uncertainty about the structure of the bottom of the array that was apparent in trials 31-40 of June 26 appeared in trials 1-10 of June 27, while the middle left to bottom right diagonal was firmly included in the searching. Cross connection was seen to occur between the lower right to left and the upper right to left diagonals. Trials 11-20 of this day(figure 4.2(f)) showed a concentration in the upper right quadrant. There was also a steady progression towards an economy of errors visible in the simplification of the trajectories of searching. Trials 21-30(figure4.2(c)) illustrate further filling in of the diagonal links. Trials 31-40(figure 4.2(d)) are at odds with their predecessors. The second row extreme right hand group of filled feeders have been excluded entirely from visitation. It is not clear why there was a sudden departure from an othewise linear progression of discovery.

The first 10 trials on the third day(figure 4.2(i)), were surprising. During the 17.3 intervening hours the bird had somehow corrected the positional errors in its searching from the last 10 trials as well as more firmly establishing the diagonal cross connections. This is an excellent illustration of how quickly the spatial memory mapping that these birds display can be revised, with or without concurrent stimulus, and then reapplied. Trials 11-20(figure 4.2(j)) display a switch in concentration in the lower left hand diagonal to the upper right. After trial 20 the experimental design was changed to that shown in figure 4.2(k), to see whether or not the bird's search was associated with the filled feeder geometry or simply a collection of unpatterned moves. If the latter were true, there would be no change in the success rate since the bird was not relying on the memory of the checkerboard pattern for guidance; if it was relying on the memory of the pattern, there would be a marked drop

in the selection of filled feeders, and this was found to be the case (Sutherland 1986). As a result of this change in design, an interesting change took place in the bird's search strategy. For the first 10 trials after the change(figure 4.2(k)), the bird, uncertain as to what exactly had been changed, reverted to an almost identical searching pattern to that displayed in the trials 11-20 of the first day(figure 4.2(b)). The last 10 trials after the change illustrate a remarkable recovery in success rate, indicating that the bird was now adapting to the new design.

4.6 Creating Movement Stereotypes to Assess Learning Patterns

Having visually taken a tour through the actual movements of all of the foraging sequences for the first bird, the next step is to try and categorize what has taken place. Referring back to the system control in figure 4.0 one can see that MicroEmacs, AutoCAD and the script generation controls have been utilized. If another level of comparison is desired, the control would now be passed to the warping algorithms, spectral analysis, and correlation techniques(Chapter 5). For the moment these are not of premier importance since the analysis is still at the gross level of detail. The control is now going to be switched from a purely numeric mode to a symbolic mode in an effort to define those structures that can be defined as search pattern atoms and rule fragments.

The expert system shell controls the manner in which the various sub-control algorithms will work. However, at this stage it is not yet clear exactly how this top level control should be designed. In Chapter 3 an interpreter was developed for categorizing the various search types. Here we categorize what has been learned from the animations into this symbolic format. Referring to figure 4.3(a), it can be seen that hierarchical arrangements of what have been labelled 'simple' searching patterns are presented. The segment bounds for each are indicated in the plots.

The segment bounds will then be used to categorize the gross level searching behaviour in the early stages of the experiment. The **upper_diagonal** and **lower_diagonal** search types are labelled as 1a and 1b. The **upper_half** and **lower_half** are labelled as 2a and 2b. The **upper_lquad**, **lower_rquad**, **lower_lquad**, **upper_rquad** are labelled as 3a, 3b, 3c, 3d respectively. Another level of complexity of searching is defined in figure 4.3(b). If a 'simple' strategy is instantiated by finding that the bird is concentrating its efforts in one section of the grid, then the appropriate level of 'complex' strategy is searched for among those shown. Since all of these are horizontal,

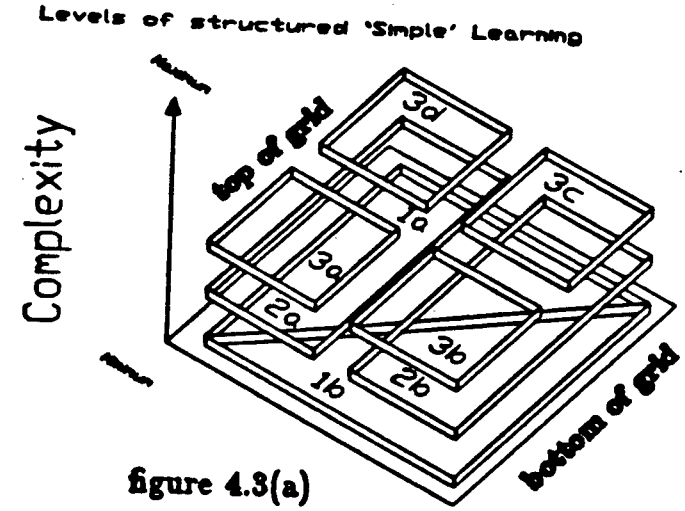


figure 4.3(a)

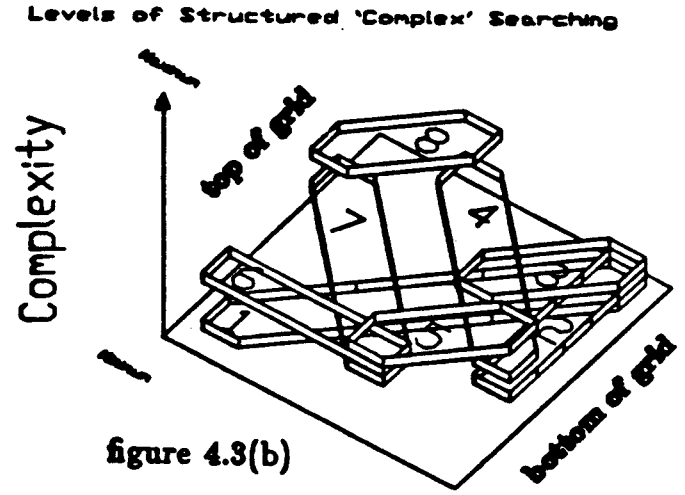


figure 4.3(b)

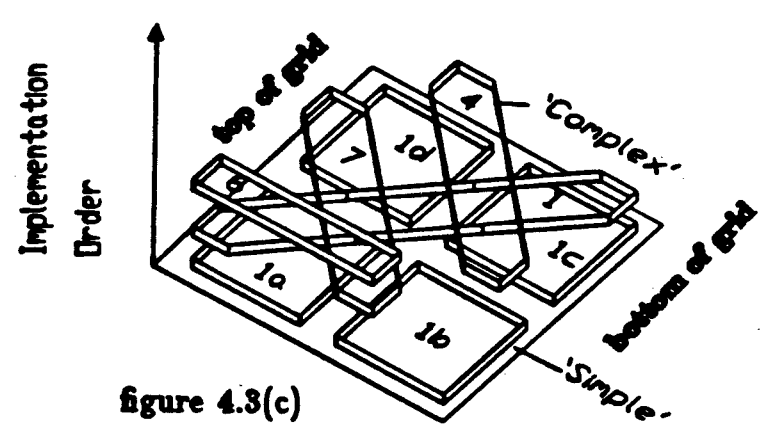


figure 4.3(c)

figure 4.3(a,b,c)

vertical, or diagonal segments, they must fall into one of the types of searches which are enumerated from 1 through 8(figure 4.3(b)). The technique for finding these instances from the data base requires two levels of translation. First, the matrix of feeder co-ordinates, e.g., 1,1 or 3,4, are mapped into a pattern with a name, such as **btm_quad**, which was discussed in detail in Chapter 3. This is done using a Prolog implementation of the idea behind Sperner's(1928) lemma that was introduced in Chapter 3. This algorithm will systematically categorize all maximal subsets of contiguous movements into their symbolic formats, each one being a named type of search.

The order in which these will then be tried is illustrated in figure 4.3(c). Having done this, the foraging sequence is then reanimated, but this time the elements of movement are not individual moves, but rather groupings of individual moves that have been found by the maximal subset procedure, and are presented as a bar on the screen. In the top left hand corner of the screen the name of the pattern is printed for identification. Each of these patterns, e.g., **btm_quad**, has a sequence of pointers to other parts of their frame components. For example, **btm_quad** has a slot that contains a procedure to establish what the numerical criterion for "high success" is, and another to establish what "complex" is, and so on. This means that for **btm_quad** to be true all of this information has to be found to be true, including its starting and ending position. As more of these patterns were found, a network of smaller, inter-related movement patterns began to unfold.

At this stage of the analysis only the animation of the raw data and the first level of patterns outlined in Chapter 1 have been analyzed. The second and third levels of pattern analysis are contained in the groups of smaller movements alluded to in the preceding paragraph. At this point it is necessary to name these patterns in a similar manner to **btm_quad**.

Maximal subset patterns can be matched to any level of generalization. These smaller patterns of movements were composed of directed searches involving edges, rows, columns and diagonals. The edges had four categories: top-right, top-left, bottom-right, bottom-left. The columns had two categories: up-column, down-column. The diagonals are first categorized as having positive or negative slope and then sub-categorized as up or down. The complete set of these movements as they are labelled is shown below:

1. **left along a top edge**
2. **left along a bottom edge**
3. **left along a row**
4. **right along a top edge**
5. **right along a bottom edge**
6. **right along a row**
7. **up a left edge**
8. **up a right edge**
9. **up a column**
10. **down a left edge**
11. **down a right edge**
12. **down a column**
13. **down a positive slope**
14. **up a positive slope**
15. **up a negative slope**
16. **down a negative slope**

When any one of the maximal subsets matches one of these movements, the starting coordinate pair is saved along with the set's length; for example, if **right along a row** has been instantiated(figure 4.3(b)), its starting location can be used as a higher level cue to one of the gross level behaviours, say, **upper_rquad**(figure 4.3(a)). Its length will then determine whether **upper_rquad**

was the only gross level pattern, or whether **upper_lquad** was also instantiated. By combining contiguous sets of these simplest structures one can ascertain whether an **upper_half**(figure 4.3(a)) movement was instantiated. Here again is an example of the inverse hierarchy of the movement taxonomy progressing from the finest level resolution to higher order concepts and those higher concepts chaining backward until a movement type is found to be true.

4.7 Creating Rules to Learn Other Rules

The simplest way combinations of rules can be recognized as another, new rule, is by simply counting up the number of times a particular rule combination occurs in a period of time, checking this number against a pre-defined frequency threshhold, and then writing this combination into the data base as a new rule by itself. Similarly, groups of these combinations can be represented as another type of new rule. Automating this process implements an elemental type of learning. During the course of analysis it was found that it was better to have the **find_new_rule** predicate report its evidence to the screen in the form of an attribute string. This string would contain the statistics of where the particular combination had occurred during the time course of the experiment, and which other birds had used the same grouping. Having examined this information the experimenter then answers "yes" or "no" to have this new rule added to the existing data base. If this process is automated so that the experimenter is not directly involved, a large number of erroneous new rules are found and added to the data base.

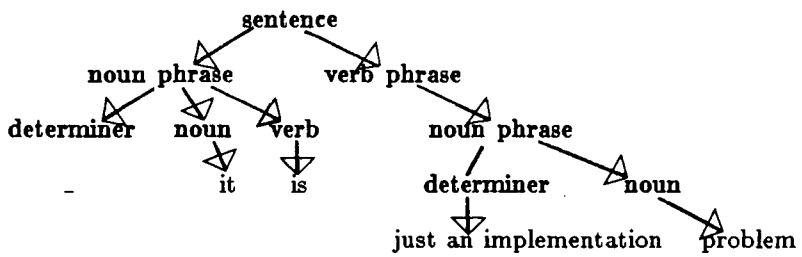
4.8 Searching for Movement Combinations Using a Grammar

Having developed the Prolog predicates, first to produce movement instances from a data base and then to recognize from its starting and ending points higher level concepts, it was natural to ask: What is the minimal combination of these simplest movements that accounts for most of the observed behaviour? That is, from the list of 16 movement types, how does one combine contiguous sets to account for most of the observed patterns in the foraging movements? Stated in more formal terms; if one were to construct a programming language to simulate hummingbird behaviour using Prolog, what are the simplest elements that would be required? Just as Basic, Fortran, Pascal, C, Lisp and other programming languages have their simplest set of rules for processing commands, one would

like to develop standardized rules to represent spatial array learning by hummingbirds. In order to understand this proposed computer language, or for that matter any language, one must begin by understanding its grammar (Chomsky, 1963). The structure of a language relies on sequences of words ordered according to specific rules. In a natural language setting these rules oversee tense, gender, voice, case, agreement and structure. These rules also apply to the descriptions, limitations, and qualifications of words and phrases(Chomsky, 1963).

4.9 Context Free Grammars

There are 4 classes of grammars known to linguists, each having a particular power. One of the very simplest of these is the context-free grammar(CFG)(Chomsky, 1963). This grammar will allow one to analyze,in this case, a contiguous set of rules and make a tree structure from them. By doing this one can generate a hierarchical relationship among component pieces. By way of illustration, a sample natural language sentence is parsed using this CFG. The sentence "it is just an implementation problem" breaks down as follows:



This context-free grammar can also be set up so that it can be coded in a computing language like Prolog. One such formulation is called a definite-clause-grammar(DCG) and expresses the context-free rules as logical statements. Using a string of foraging rules to make a sentence about what has happened will permit prolog to simply parse them in a manner similar to the above example. The problem of parsing a string of this pseudo-language now becomes the problem of proving that a theorem follows from the language's definite clause axioms. The DCG's break a sentence down into a noun phrase and a verb phrase. A DCG rule looks like the following:

Sentence −− > noun_phrase, verb_phrase.

One reads the $--->$ to mean "can take the form of" and the comma to mean "followed by". Applying this to the sentence one obtains "A sentence takes the form of a noun phrase followed by a verb phrase."

The same statement in prolog is

sentence(A0,A) :-

noun_phrase(A0,A1), verb_phrase(A1,A).

This sentence reads as "A sentence extends from A0 to A if there is a noun phrase from A0 to A1 and a verb phrase from A1 to A."

The prolog statements to perform this parsing are extremely simple and easy to enter. For example, the sentence **is the bird using [6,12].** can be typed in as is. It is then translated by the parser into the following prolog predicate:

sentence([is,the,bird,using,([6,12]],X).

Another prolog clause will fill in for **[6,12]** , **"right along a row and down a column"** from a list of rules that has been previously read in from a file. This declaration's noun phrase " **the bird** " is separated from its verb phrase "**is using [6,12]** ", and then " **[6,12]** " is further parsed to provide input to the **search_strat(List,X)** predicate. Here the theorem prover will try to instantiate **X** = **[6,12]** from **List** . **List** is the data base containing the sequences of the sixteen enumerated rules that have been located sequentially in some section of the foraging sequence. This very simple DCG provides a more natural language interface to the rule data base query system as well as returning the complete parse tree for each sentence. A complete parse tree for all data samples in **List** for all search strategies **[6,12]** that can be found in the body of rules denoted as **Rules**, can be obtained with the **follows_from** predicate developed in Chapter 3. It would be called with the following arguments:

follows_from(search_strat(List,([6,12]),Rules)).

4.10 Applying a DCG and Movement Rules to Foraging Sequences

A natural language interface to the prolog predicates is a useful aid in bypassing the syntax required to make a system query. The DCG is the simplest of grammars and is very flexible. It can be quickly expanded with other parsing predicates to present any aspect of these rule combinations that are found to be important. As will be shown the need to expand becomes obvious when the grammar, rules and maximal sets are applied to actual foraging sequences.

The first test for the maximal subset finder was conducted on the movements of the first bird during the first 10 trials, from June 26, 1985. Having read the 353 coordinate movements into **List** , the **max_subset** predicates began searching for instances of any one of 16 rules contained in the list **Rules**. When an instance was found, the rule was written out as a string. Its starting coordinates and length were written out to another file for later use with the DCG. As rule instances were located, their string names were also written to the screen in the upper left hand corner while the start location was plotted on a 8×8 grid in the center of the screen. Blocking these individual movements into rules and then animating them proved to be extremely useful for visually recognizing patterns of rule usage. This compression into rule fragments simplifies the visual patterns that were previously viewed on an individual movement basis. The additional visual reinforcement of having the rule type echoed on the screen enhanced further the process of recognizing groups of patterns that may be included as part of some foraging strategy. Subsequent sets of ten trials for the rest of June 26 were processed, then the rest for June 27 and 28.

As the bird learned the boundaries of the experimental design another feature was observed from the rule pattern animation. This first bird cycled the **down a negative slope** top to bottom and then looped back around in an approximation to a figure eight. This was not apparent in either the raw data animations or from the one-dimensional warpings and proved useful for the next level of pattern recognition.

4.11 Searching for Minimal Sets of Rule Combinations to Form a Searching Strategy

The last stage in the symbolic recognition was to find those contiguous rule combinations that accounted for most of the behaviour. This yielded more surprises. The correct combination of these sets was expected to be complicated and to involve many partial sequences that somehow would have to be integrated into a composite structure. This was not to be the case.

It should be noted here that the spatial resolution in the analysis to date was at the sector size. Analysis of movements at a smaller scale will be dealt with in Chapter 5.

The first bird's first ten trials were very simple and were composed mainly of **right or left across a row, and up or down a column**(figure 4.2(a)). This accounted for around 60% of the total movements. Another 30% was accounted for by **right across a bottom edge** and **down a left edge**. The remaining 10% was made up of individual rules. By the end of the first day(figure 4.2(d), **up or down a negative slope** had been added in connection with **right across a bottom edge**.

On the second day the simple row column search was incrementally replaced by positive and negative diagonal searches and by the end of the second day the diagonal stereotype in its various forms was the dominant search strategy(figures 4.2 e through l).

Along with this searching came the figure-eight searching cycle of simple movements. If one was to take the grid and stand it on the lower left corner(location 8,8 in figure 4.3(d)), a figure eight could be drawn whose upper half encompasses the filled sectors 1,3,6 and 9. Its lower half would encompass filled sectors 8,11,13,16 and cross at feeder location (4,4). The bird's strategy could then be described simply as: use a negative, or positive slope diagonal from an edge until the lower corner is reached, then loop upwards in either a clockwise or counter clockwise manner until a top edge is encountered and then begin again. In terms of the rules, the search would consist of **right along a row** and **down a column**, then **right along a row and up a column**, then **left along a row** and **up a column**. The result is a box-like search of the quadrants. A more sophisticated search would use **up or down a positive or negative slope**. This results in an 'X' crossing at location 4,4 and two diagonals connecting the ends of the 'X'. This is roughly the manner in which this cycling was actually done by the birds, as observed in the animation of movement rules.

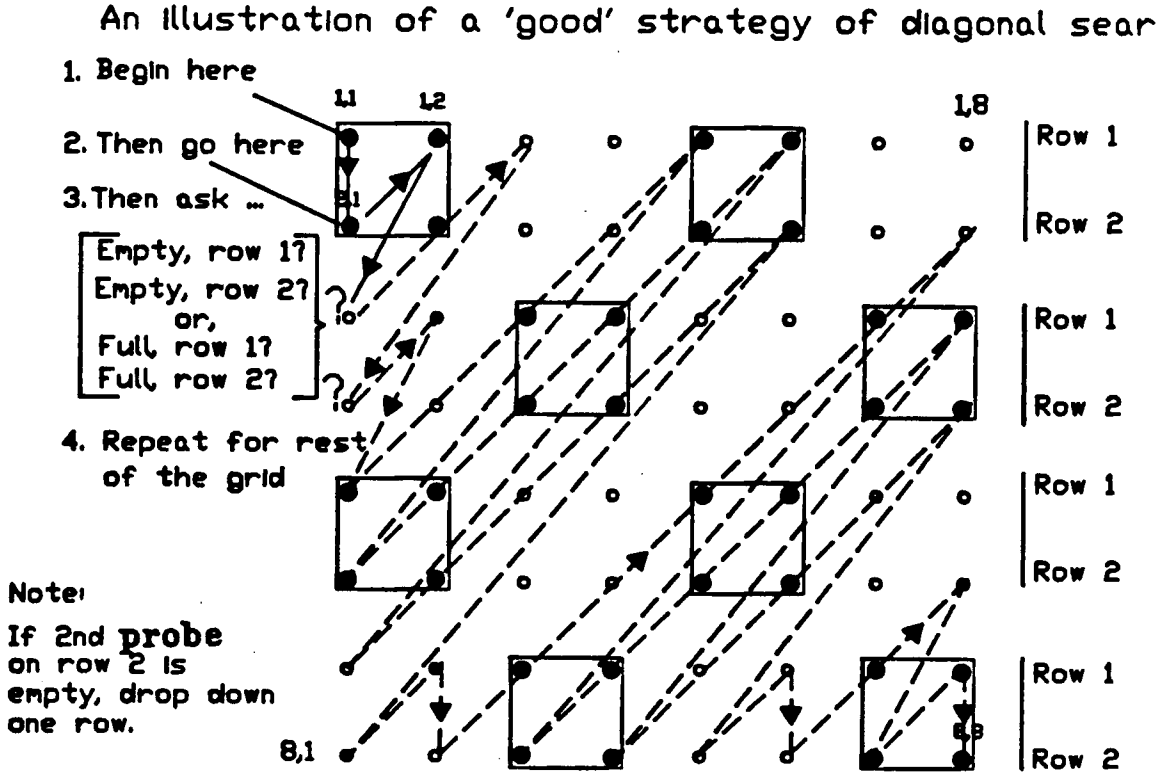
There are many such candidates for ways of linking these rules together to form a searching strategy. The Prolog environment makes their implementation a simple task. Ultimately, the experimenter would want many of these strategies present in a knowledge base that is used strictly for testing strategies. Typically, this knowledge base would be composed of predicates that describe how a strategy is to be implemented. A prototype prolog predicate was developed to provide a template for future development. Its form is as follows:

search_strategies(Rule_combo_list,Rule_instance_file,Is_true)

Note: this predicate would be one of many that procedurally represent searching strategies.

The list **Rule_combo_list** would contain those combinations of rules to be used in some user specified combination, plus the mechanics of searching to be used. The **Rule_instance_file** would contain the previously animated rule instances. The **Is_true** string would return the % success rate of using this strategy as well as any other desired statistics. Through this knowledge base format it is now possible to code up as many strategies as the experimenter wishes to test, and then proceed sequentially down the collection of these predicates trying to verify their existence. As a demonstration of how such a strategy would be implemented a **Rule_combo_list** was composed and was given the name **good_diagonal_search**. The mechanics of this search strategy and the rules that would be used are illustrated in figure 4.3(d).

This search was discovered by closely observing the individual birds correct mistakes on successive passes across the grid. All of the birds tended to start in the first sector and generally proceed as shown in figure 4.3(d). In terms of rules used, **down a column**, and **up, or down a positive slope** were the primary candidates. Combining these with the error correction indicated in the figure yields a success rate close to the birds' general performance. The predicate begins with grid element (1,1) and is continued until (8,8) is probed. There are many orientations of the 'good' strategy. Starting points other than the one indicated in figure 4.3(d) yield different success rates. By moving the starting point of the algorithm to different positions a wide variety of results can be obtained. The birds appeared to use this strategy of moving the starting point of a search and then using the diagonal search algorithm.



Rules required for instantiating a 'good' search strategy.

1. up, or down a positive slope
2. down a column
3. both of the above in some combination

figure 4.3(d)

4.12 Some Selective Differences Among Birds in Learning an Experimental Design

Having garnered a strategy of rule utilization and sequencing from the first bird, the next test of the knowledge representation system was to take this strategy and try to apply it to subsequent individuals. This provided another surprise concerning the way these organisms adapt to a spatial pattern. The second individual's first 10 trials were first animated, then parsed into maximal subsets and finally searched for minimal rule set combinations. The way in which this individual set about learning the diagonals was entirely different from that of the first individual. Figures 4.4(a through d) are for the the first day; figures 4.4(e through h) are for the second day and have the same trial blocking as the first bird's behaviours. These illustrate differences in the way individuals learn a spatial pattern.

This second invividual began by using **right or left along a top edge** and **up or down a left edge**(figure 4.4(a)). It did not venture further than the first diagonal. During the next 10 trials(figure 4.4(b) this strategy was completely abandoned and the search resumed much in the way that the first bird set about establishing the design boundaries. The difference between the first and second bird was that the second tended to use the edge rules and work inward, whereas the first individual simply used a row and column strategy. This was found to be the case through the next 10 trials. During the last 10 trials(figure 4.4(d)) exactly the same rules that had been used for the first ten were implemented but in the opposite diagonal. The only differences were that the (8,8) coordinate was used as a starting point rather than (1,1) and the edges were the primary clues.

By now it is clear that there are a great variety of ways in which a simple rule base such as the one used here can be implemented. In fact the ways in which this knowledge representation system can be tested are also very large. The surprise between these two sets of rule implementations can be seen in the success rate as shown in figures 4.5(a through e). In contrast to the distinctly different movement strategies, the similarity in the overall success rates of birds 1 and 2 is surprising (figures 4.5(a,b)). Success is measured as the number of probes at filled feeders over the total number of feeders probed, multiplied by 100 that took place during a trial. This measure is in contrast to that used by Sutherland (1986), who considered revisits to empty feeders as being successful. The variation in the ways these birds were successful further illuminates that a flexible DCG is needed

trials 1-10

figure 4.4(a)

trials 21-30

figure 4.4(c)

Animation of the foraging path
for bird 2, June 26, 27
blocked into 10 trial increments

June 26

trials 11-20

figure 4.4(b)

trials 31-40

figure 4.4(d)

figure 4.4(a,b,c,d)

figure 4.4(e)

figure 4.4(g)

Animation of the foraging path
for bird 2, June 26, 27
blocked into 10 trial increments

June 27

figure 4.4(f)

figure 4.4(h)

figure 4.4(e,f,g,h)

An Illustration of the success rate
of all 5 birds for June 26, 27, 28

Bird 1

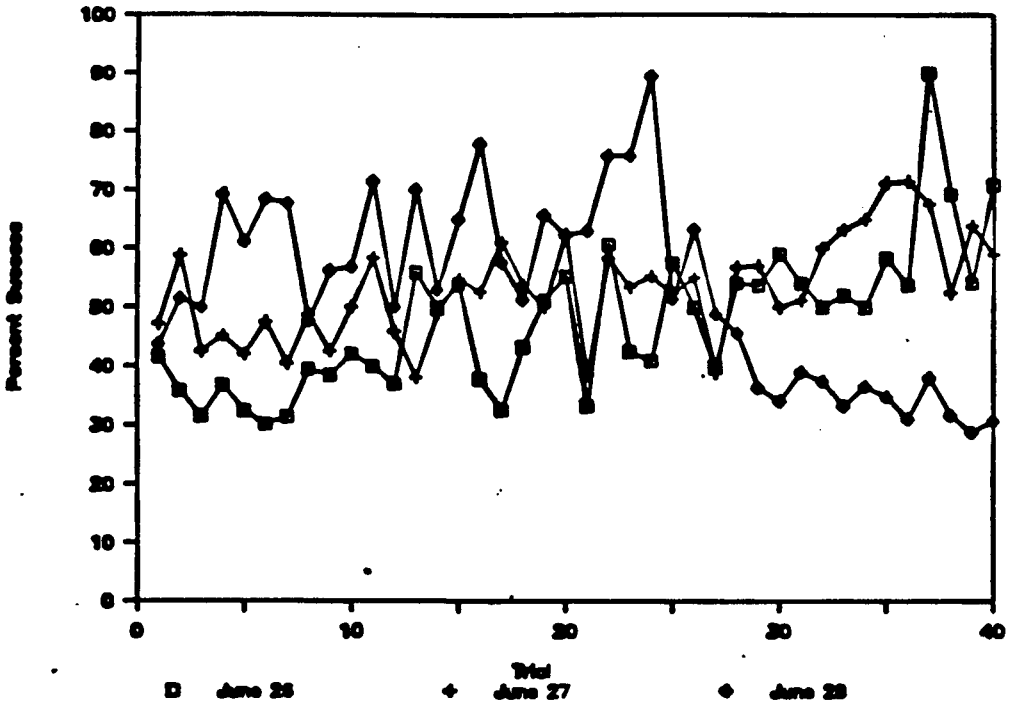


figure 4.5(a)

Bird 2

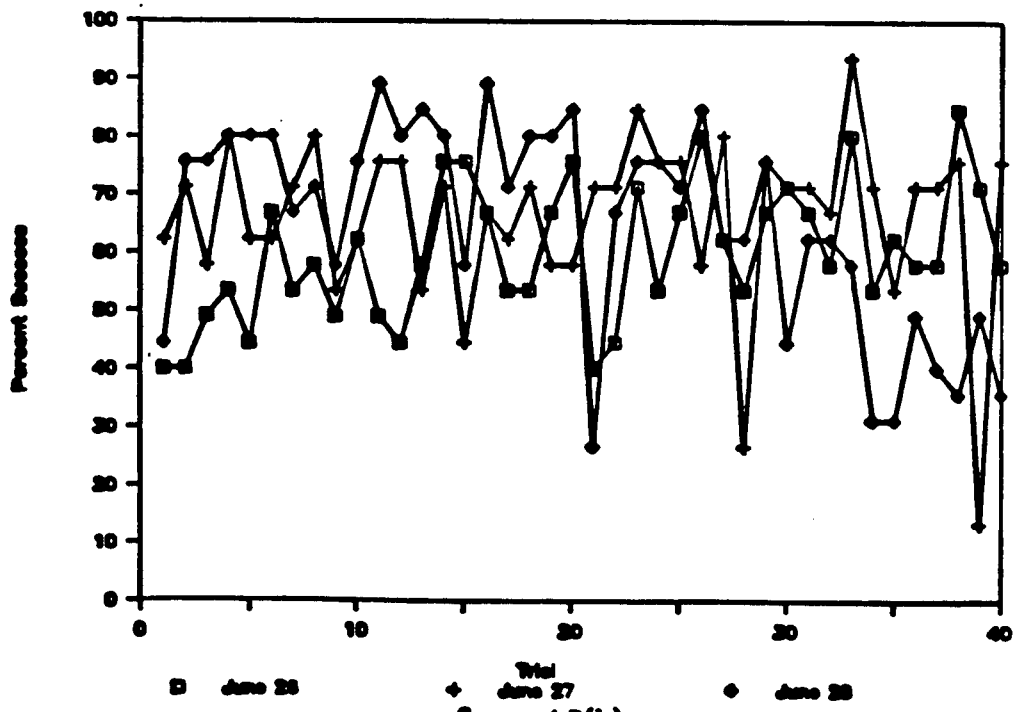


figure 4.5(b)

figure 4.5(a,b)

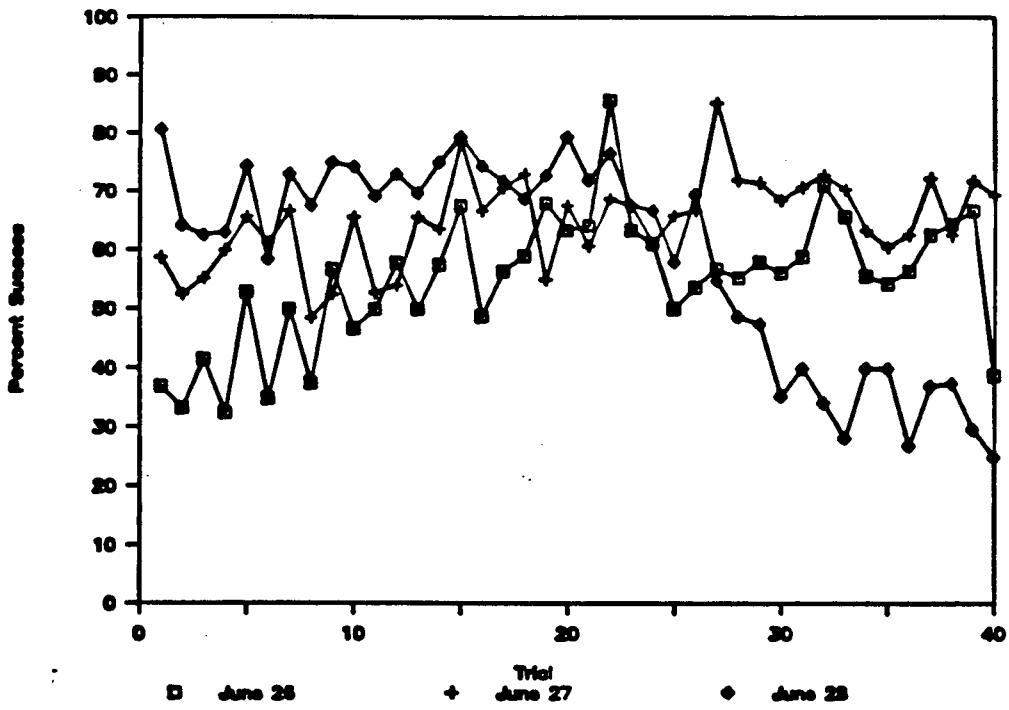


figure 4.5(c)

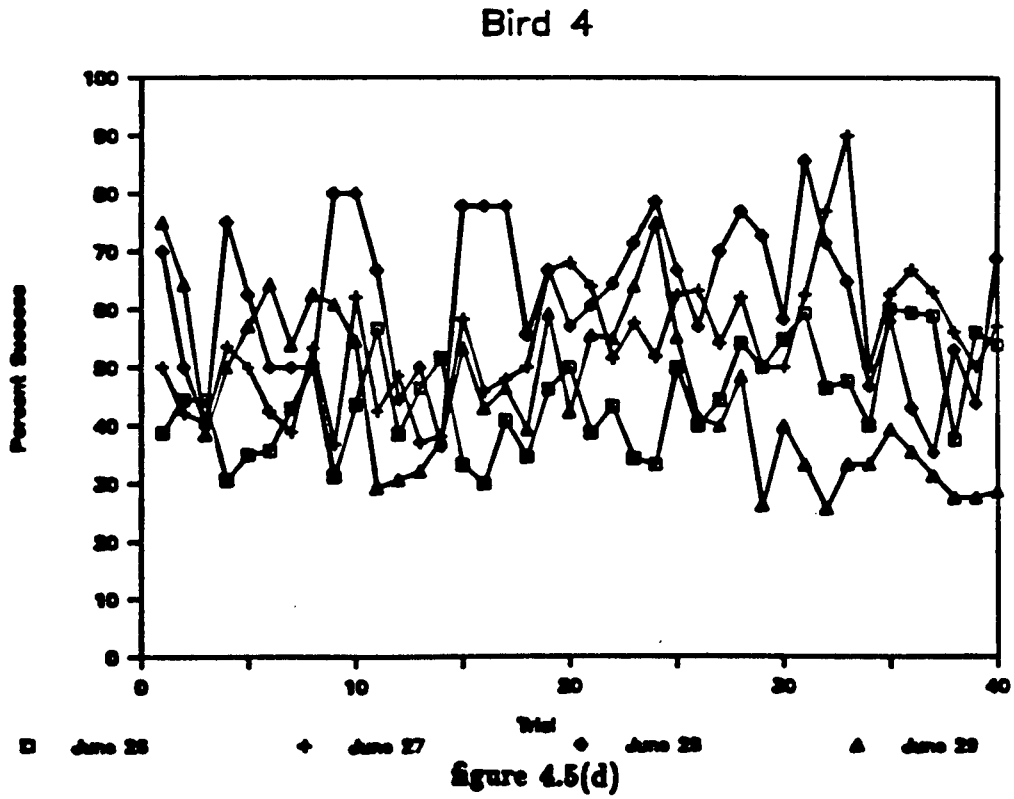


figure 4.5(d)

figure 4.5(c,d)

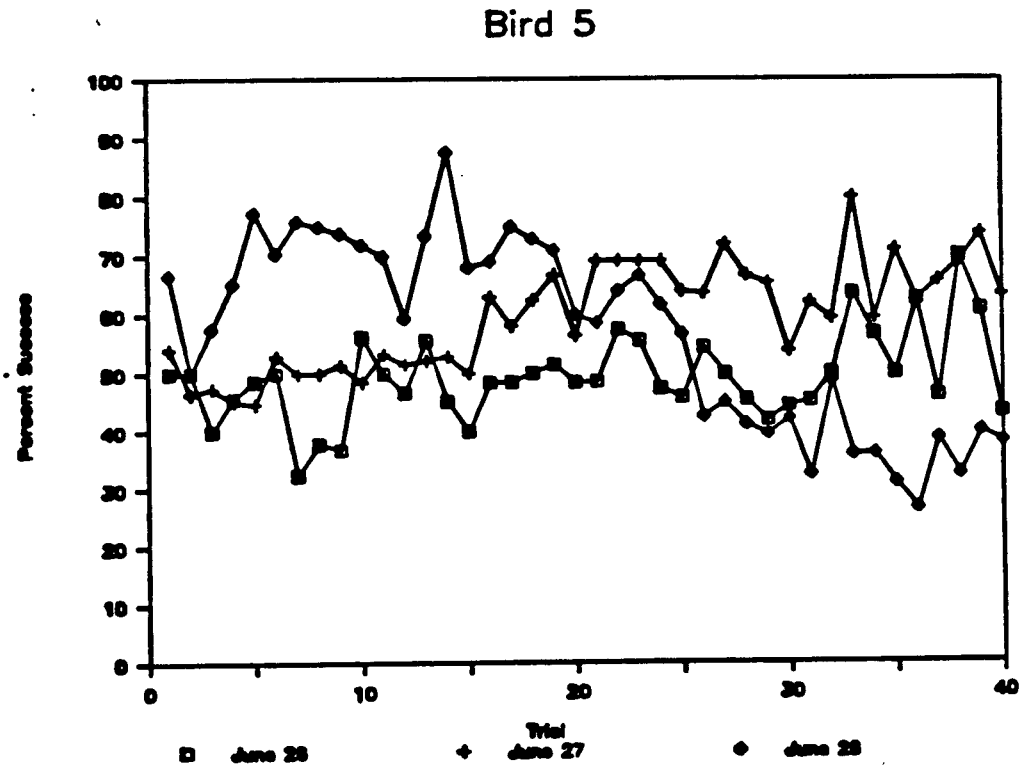


figure 4.5(e)

figure 4.5(e)

to accommodate individual variations in strategy. It also serves to illuminate the failings of the gross level categories of what is 'simple' and what is 'complex'. The other two days of the second bird's foraging were simply refinements of what has already been demonstrated on the first day. Beginning at an edge and working inward seemed to be this bird's way of establishing the design pattern. Another observation that applies to both birds was the tendency to retain a pattern that was only partially successful while searching for the rest of its structure. This became visible when viewing the animations when the design was switched on the 25th trial on the third day. Initially after the change(figures 4.2(k,l)), the old pattern was observed to be retained, while the new pattern was learned. The learning for the new pattern was begun in the same manner that was used for the first 10 trials of the first day. This was true for all five subjects.

The third individual's strategy was a combination of the first two. See figures 4.6(a through k).

The third bird came closer than any individual to implementing the 'good' algorithm; it began the trial by entering the grid at the location illustrated in figure 4.3(d) and tested the sectors in roughly the same manner as in the figure. The diagonal organization was observed to be established during the first 10 trials and the bird's performance was consistently higher than that of any of the others. It was observed to systematically put together those rule elements required to establish the boundaries and then proceeded to eliminate those that were not rewarded. After the design was switched the bird did not eliminate the previously learned design as quickly as the others did.

The fourth individual was tested for four consecutive days. The figures 4.7(a through h) are for the first two days only.

Its behaviour was erratic in comparison with the others but bird 4 had similar success rates. The edges again provided most of the design clues, but an edge was never completely searched and many trials consisted of as few as six feeder probes. This meant that the overall number of successes was much lower that the others; but based on the percent success rate, it compared favorably. The bird also had a much simpler minimal set of rules composed of individual edges, rows and columns each being applied separately rather then as a contiguous strategy.

The fifth individual used a strategy similar to the first individual. Figures 4.8(a through h) represent the first two days. Figures 4.8(i and j) represent trials 21 to 30 and 31 to 40 all of which were undertaken on the third day. Here again the intervening period between the first day's trials and

trials 1-10

figure 4.6(a)

trials 21-30

figure 4.6(c)

Animation of the foraging path
for bird 3, June 26, 27, 28
blocked into 10 trial increments

June 26

trials 11-20

figure 4.6(b)

trials 31-40

figure 4.6(d)

figure 4.6(a,b,c,d)

figure 4.6(e)

trials 1-10

trials 21-30

figure 4.6(g)

Animation of the foraging path
for bird 3, June 26, 27, 28
blocked into 10 trial increments

June 27

trials 11-20

figure 4.6(f)

trials 31-40

figure 4.6(h)

figure 4.6(e,f,g,h)

trials 1-10

figure 4.6(i)

trials 11-20

figure 4.6(j)

figure 4.6(i,j,k)

Animation of the foraging path
for bird 3, June 26, 27, 28
blocked into 10 trial increments

June 28

Note: the filled sectors
have been shifted
1 sector to the right
for this set of trials

trials 21-30

figure 4.6(k)

figure 4.7(a)

figure 4.7(c)

Animation of the foraging path
for bird 4, June 26, 27
blocked into 10 trial increments

June 26

figure 4.7(b)

figure 4.7(d)

figure 4.7(a,b,c,d)

figure 4.7(e)

figure 4.7(g)

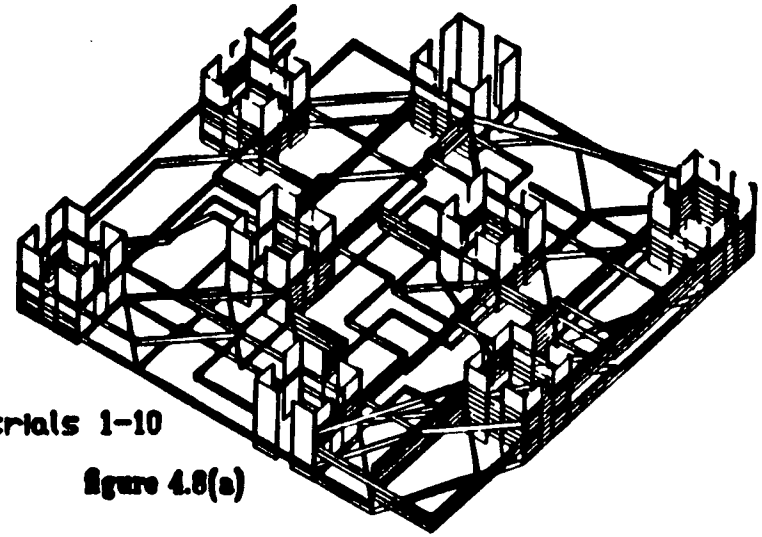
Animation of the foraging path
for bird 4, June 26, 27
blocked into 10 trial increments

June 27

figure 4.7(f)

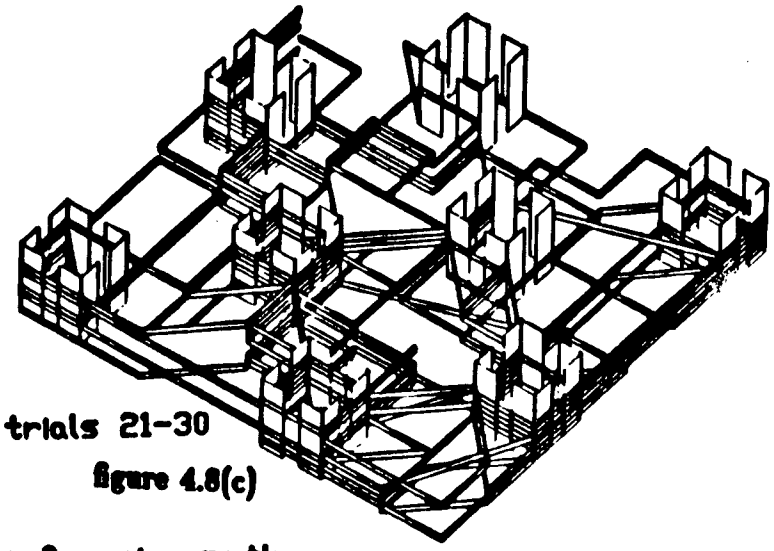
figure 4.7(h)

figure 4.7(e,f,g,h)



trials 1-10

figure 4.8(a)

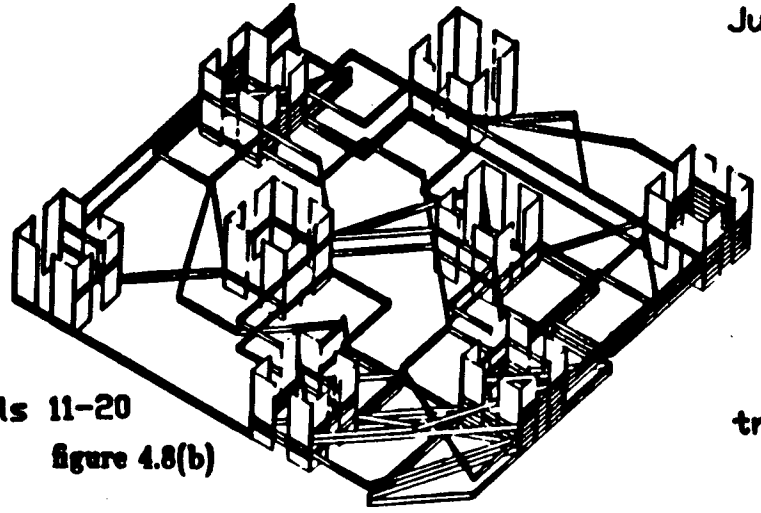


trials 21-30

figure 4.8(c)

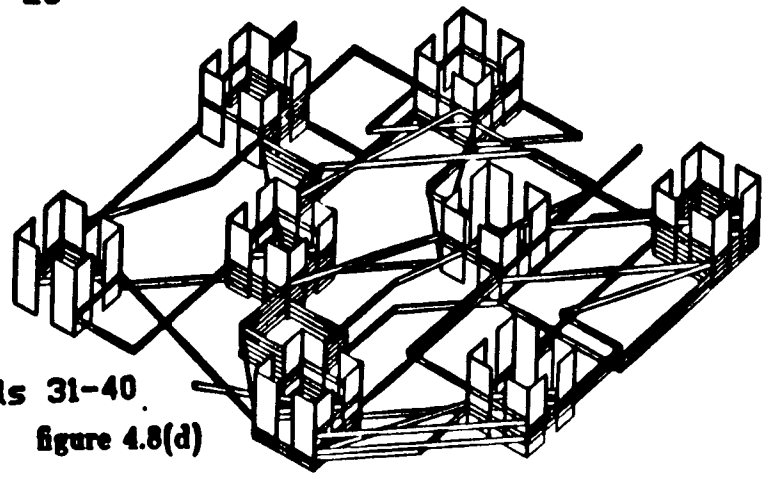
Animation of the foraging path
for bird 5, June 26, 27, 28
blocked into 10 trial increments

June 26



trials 11-20

figure 4.8(b)



trials 31-40

figure 4.8(d)

figure 4.8(a,b,c,d)

trials 1-10

figure 4.8(e)

trials 21-30

figure 4.8(g)

Animation of the foraging path
for bird 5, June 26, 27, 28
blocked into 10 trial increments

trials 11-20

figure 4.8(f)

June 27

trials 31-40

figure 4.8(h)

figure 4.8(e,f,g,h)

Animation of the foraging path
for bird 5, June 26, 27, 28
blocked into 10 trial increments

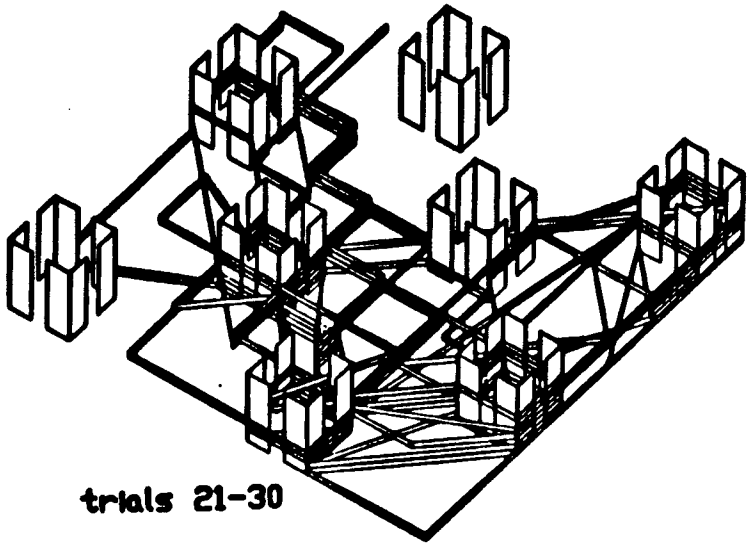
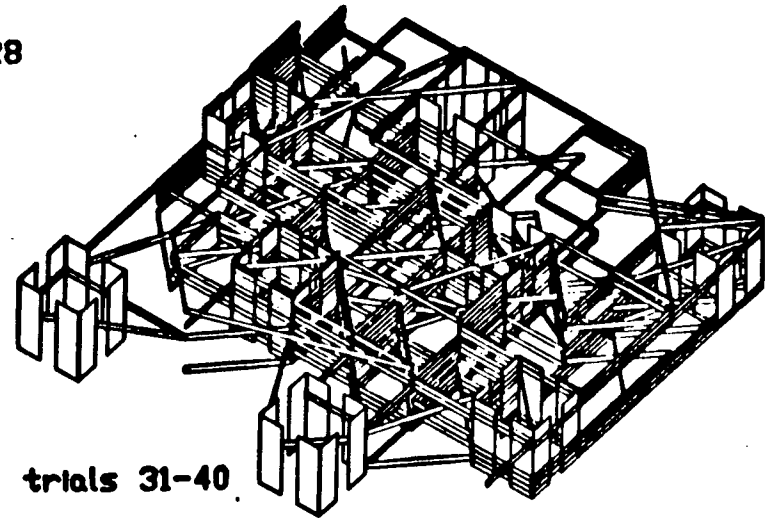


figure 4.8(i)

June 28



Note: the filled sectors
have been shifted
1 sector to the right
for this set of trials

figure 4.8(j)

figure 4.8(i,j)

those of the second day provided this individual with the opportunity to connect the top diagonal pattern with part of the internal structure of the second diagonal pattern. Here the first 10 trials on day one used **right or left across a row** and **up or down a column** about 60% of the time, and individual rules the rest of the time. By the end of this first day the **bottom edge** rules had been added and accounted for about 20% of the movements. When the design was switched it did not exhibit much change in strategy until 10 trials later. By the end of the 40 trials it had not uncovered much of the new design.

4.13 Summary of Chapter 4

- A knowledge representation system for foraging behaviour was implemented on an IBM AT microcomputer.

- The system control design, the graphics window and the editor were described.

- Scripts for animating the foraging behaviour were developed and applied to the data bases.

- A maximal set of rule stereotypes were developed and searched for in the data bases.

- A very simple DCG was developed to find rule combinations.

- Minimal rule subset combinations were searched for in the data bases.

- A typical 'good' diagonal search strategy was developed, implemented and tested on the output of the maximal subset algorithm.

- The 'good' diagonal search was shown as a prototype for a knowledge base of search strategies.

- The selective differences in the application of these rules among the five individuals tested were discussed.

Chapter 5

5.1 Learning Spatial Patterning Using a Knowledge Base

The numeric, graphic and symbolic algorithms used in this thesis have one function in common: they provide rational tools that proved to be effective in unravelling structural relationships existing in a collection of behavioural propositions. These in turn were used as a series of tests of a knowledge representation system for hummingbird foraging in a laboratory environment.

The next aspect of the system investigated was the frequency in use of pairs of these simple propositions, such as **right across a row** and **down a column**. From these it was expected that the regularity of their use could further illuminate the differences in the types of strategies used by the birds. The foraging rules developed in chapter 4 provided the basis for a type of propositional lattice based on the grouping of related rule preferences that were found to be correlated. A proposition lattice was implemented for two reasons.

The first reason is concerned with the type of data that were analyzed in this section. Normally two kinds of data are obtained in an experiment. The first kind is the type whose magnitude has meaning. The second kind is such that only the relational values of the components are important. The patterns, such as **right across a row** are of this second type.

A proposition **P(a)** that object **a** satisfies predicate **P** means, in the language of pattern recognition that object **a** is placed in class **P**. The idea of deriving logic from the geometrical representation of **P** was presented by Watanabe (1985). This approach can be summarized as follows:

First one decides on the geometrical representation of the propositions and then proceeds to define the geometrical representation of implication. From these definitions all geometric representations and logical implications must follow. If the correspondence between these predicates and their representations is one-to-one, then the entire logical system can be treated from this point of view. A complete discussion of this type of lattice and how it differs from the usual interpretation of propositional calculus can be found in Wantanabe, (1969) and Kleene, (1952).

The second reason for using propositions is because of the simplicity of their structure. Defining propositional relationships between rules and rule pairs makes their relationships strictly geometrical. This geometry of relationships made finding patterns of rules in the data base much faster than a

tree search would have been. With the geometry of the relationships fixed, the search for pattern matches is a simple set of if-then-else commands. In addition the order of their discovery can be fixed eliminating needless checking of combinations of orders. The form of representation chosen in this thesis is that of a propositional lattice. An illustration of this simple lattice can be seen in figure 5.0. The rules for constructing this lattice are presented in their entirety in section 5.2.

The correlation computed between these propositions was for pairs of pairs of rule use combinations. The amplitudes of the data were the frequency of selection for a particular rule during the course of a 40-trial period. This meant that by using a propositional lattice, rule combination pairs were found in the data base. Each new occurrence of this combination was noted, and the total number of times it occurred during a day, two days, or three days was saved in a file. The data in this form were now ready to be analyzed.

A sample sequence of these rules or predicates that have been discovered from a recorded foraging sequence is as follows:.

'new trial'

'right along a row'

'down a column'

'right along a row'

'down a column'

'right along a row'

'up a negative slope'

'right along a bottom edge'

'new trial'

'right along a top edge'

..... etc. ...

'eof'

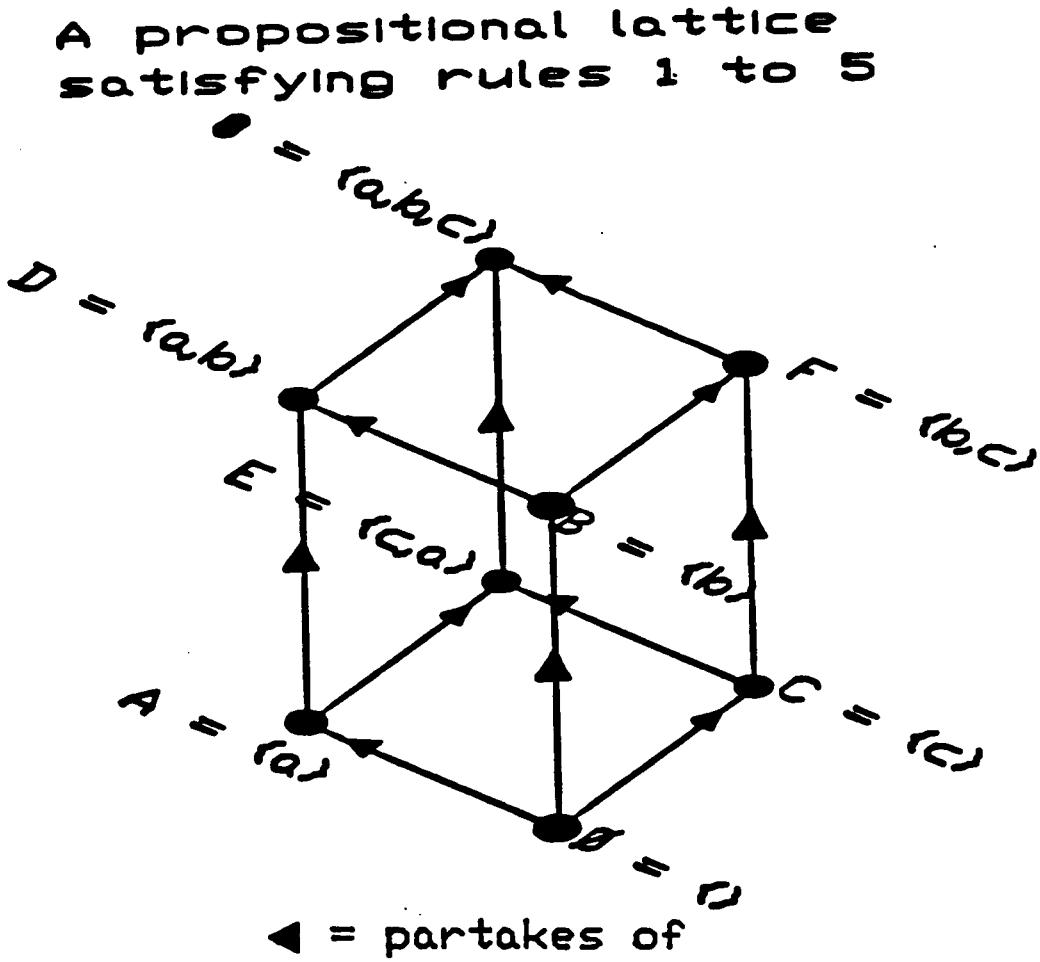

A propositional lattice
satisfying rules 1 to 5
● = [a,b,c]
D = [a,b]
F = [b,c]
E = [c,a]
B = [b]
A = [a]
C = [c]
∅ = [ ]
◀ = partakes of

figure 5.0

There are two additional predicates provided for partitioning trials, these are: **'new trial'** and **'eof'(end-of-file)** This identification procedure is repeated for each of the three days for each of the five birds. The resulting format comprises a collection of time series, one for each predicate with each containing 40 elements or observations. These are replicated for each bird three times. Each element of the time series, on any day, represents a preference by an individual bird for using a particular predicate, or rule during a particular trial. Therefore, the entire time series provides a record of changes in preference for this predicate during the course of a day's foraging. What one wishes to know is:

1. **How are these preferences interrelated as the day unfolds?**
2. **How are these interrelated between days?**
3. **How are they related between individuals?**
4. **Can anything be deduced from these related selections?**

5.2 Grouping Rule Preferences in a Propositional Lattice

The easiest way to organize this collection of predicates or rules is in a propositional lattice. The motivation for its use lies in the fact that the DCGs and treelike structures require that the transitions proceed downward until satisfied and then backtrack to the root again. When the pattern matching searches the database for instances of the eight rules and their combinations their order of occurence is of primary importance. Propositional lattices have a very simple named relational structure as can be seen in figure 5.0, and eliminate having to traverse a tree in order to be instantiated. This makes them very fast and, therefore, a natural choice for this type of pattern recognition.

A propositional lattice denoted as

$$L = \{rule1, rule2, ..., rule8\}$$

is such that it has five basic characteristics which are as follows:

1. Among pairs of elements there exists a one-way relation "rule1 partakes of rule2" which is written as

$$rule1 \rightarrow rule2.$$

2. If $rule1 \rightarrow rule2$ **and** $rule2 \rightarrow rule1$ **then** $rule1 = rule2$ and these propositions are said to be equivalent.

3. This "partaking" relation is transitive, that is, if

$$rule1 \rightarrow rule2 \ and \ rule2 \rightarrow rule3, \ then \ rule1 \rightarrow rule3.$$

4. The lattice L has a special element $\bullet$ that is called the "whole", such that any element X of L partakes of it:

$$X \rightarrow \bullet, X : arbitrary.$$

5. The lattice L contains a special element $\oslash$, called the "zero" which partakes of every element X of L :

$$\oslash \rightarrow X, X : arbitrary.$$

The characteristics 1 through 3 generate a set that is partially ordered having an upper and lower bound. If this set satisfies characteristics 4 and 5, it is called a lattice.

As an illustration of how this lattice works, consider an arbitrary combination of **right across a row, down a column** and **down a negative slope** selected from the 16 element lattice L. For the sake of brevity, re-label these as $a, b,$ and c. Suppose one wants to know all the possible groups that can be formed from these rules. There are 3 C_1^3 groups, each containing one rule, $\{a\}, \{b\} or \{c\}$; there are 3 C_2^3 groups containing $\{a, b\}, \{a, c\}, \{c, b\}$. If one adds to this the empty group $\{\}$ and the group of all rules $\{a, b, c\}$ one now has a group of eight elements.

If one states $A \rightarrow B$, this means that every rule included in group A is included in group B. If one labels the empty group $\{\}$ as $\oslash$, and the full group $\{a, b, c\}$ as $\bullet$, this set of 8 elements satisfies all three required characteristics for a partially ordered set with an upper and lower boumd. The only problem is the assertion that $\oslash$, which is $\{\}$, partakes of any other group. This can be circumvented by stating that $A \rightarrow B$ is equivalent to saying that every rule not included in B is not included in A either. This means that $\{\} \rightarrow \{a, b\}$ holds because the only rule c that is not included in $\{a, b\}$ is not included in $\{\}$, in exactly the same way it is not included in $\{a\}$ or $\{b\}$. Hence, $\{\} \rightarrow \{a, b\}$ as well as, $\{a\} \rightarrow \{a, b\}$ and $\{b\} \rightarrow \{a, b\}$. A geometrical illustration of this can be seen in figure 5.0.

5.3 Correlation Measures on Time Series of Rule Frequency Pairs

The time series pair combinations among all of the five birds for each of the three days of observation is very large. There are 120 pair combinations possible for any bird on each of three days. There are five birds and 10 pairs of cross combinations of these 120 pairs for the different bird combinations. During the course of analysis, it was discovered that the eight search patterns associated with traversing the edges of the grid could just as easily be represented by the original eight, collapsing the set to half its original length. Since it made no difference to the searching algorithm whether a pattern was included on an edge the searching process was greatly speeded up by the deletion of these spurious rules. This new subset was formed from the rule illustrated in figure 5.1.

reference number

1. **'right along a row'**
2. **'left along a row'**
3. **'down a column'**
4. **'up a column'**
5. **'down a negative slope'**
6. **'up a negative slope'**
7. **'down a positive slope'**
8. **'up a positive slope'**

figure 5.1

To further break down the problem the analysis was done for each bird separately. The rule use frequencies were then summed for three and two days. Finally, the rule use is presented for each of the three days, so as to illustrate how their use changed over time.

The expected value and the linear trend were removed from each of the series in turn and a simple correlation coefficient computed using formula 5.0.

$$\rho = \frac{\sigma_{XY}}{\sigma_X \sigma_Y} \qquad 5.0$$

where σ_Y, and σ_X are the variance of X and Y respectively, and σ_{XY} is the covariance of X and Y.

The results are in the form of an upper triangular matrix whose elements are simple correlations between rule pairings, and whose height in the histograms is the amplitude of the correlation. The pairings are denoted by their reference numbers and are illustrated in figure 5.1.

The matrix of correlations has elements numbered:

1,2	1,3	1,4	1,5	1,6	1,7	1,8
	2,3	2,4	2,5	2,6	2,7	2,8
		3,4	3,5	3,6	3,7	3,8
			4,5	4,6	4,7	4,8
				5,6	5,7	5,8
					6,7	6,8
						7,8

figure 5.2

Figure 5.3 illustrates how to read and interpret the 3 dimensional histograms. The results are presented for days 1 through 3 for birds 1,2,3,4 and 5. These histograms are presented in figures numbered 5.3(a,b,c), 5.4(a,b,c), 5.5(a,b,c), 5.6(a,b,c), and 5.7(a,b,c). The stick located at location 4,8 has a positive amplitude of 1(white), and a negative amplitude of 1(black). It is placed here as a scale for amplitude comparisons.

The correlations and their plots are not intended as a strictly statistical measure that has a significance level associated with it, rather they are presented as another type of pattern that groups the propositional uses together in order to facilitate quick visual recognition of differences among the birds' behaviours.

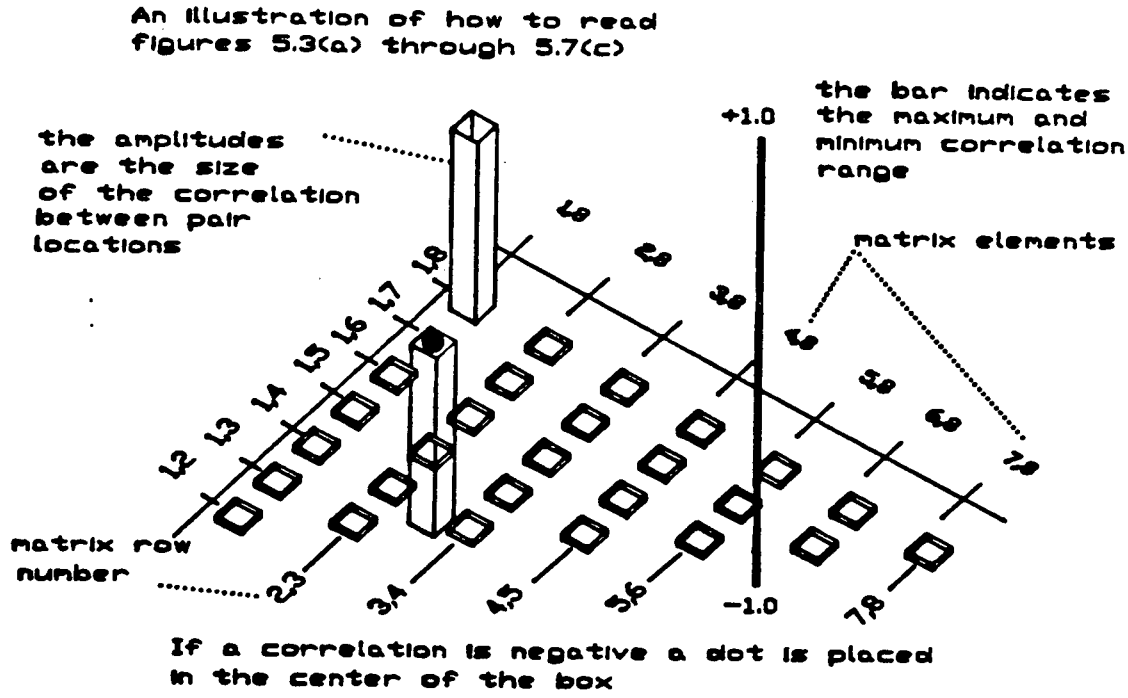

An illustration of how to read figures 5.3(a) through 5.7(c)
the amplitudes are the size of the correlation between pair locations
+1.0
the bar indicates the maximum and minimum correlation range
matrix elements
matrix row number
-1.0
If a correlation is negative a dot is placed in the center of the box
correlation between pairs of pairs of rules occurring in an 8 sec interval

figure 5.3

Examining the first bird's progress (figures 5.3 a,b,c) through the three days one can see several shifts in its learning strategy. The correlation coefficients are presented so that the first two rows are right and left row search propositions, the next two are column propositions, the last three are slope propositions. A predominance of propositional pairs that have rows as the first proposition indicate that the diagonal structure has not been discovered and that a simpler level of pattern recognition is being applied. On the second day the connection of the slope propositions as information carriers can be seen in the inclusion of propositions 1,4, 1,8, 2,8 and 3,8. These pairs replaced with their labels are **right along a row** and **down a column** , **right along a row** and **up a positive slope** , **left along a row** and **up a positive slope** , **down a column** and **up a positive slope**.

The algorithm proposed in Chapter 4 as the 'good' search pattern is executed using these propositions. For details see section 4.11 in chapter 4. It should be noted that the correlation plots were used as visual guides when the patterning sequence program was re-run. This permitted visual focus on the prominent propositions indicated in the correlation plots. The **up and down column** propositions indicate that the figure-eight searching observed in the animated sequences were being used in conjunction with some 'bird' version of the 'good' search. One other feature became apparent as these preferences were observed through time. The birds generally acquire these searching rules fairly quickly, but will not generally give them up. It is perhaps possible that the repertoire of responses is retained for some not-yet-understood reason as something to fall back on if things are not proving successful. This implies that spatial patterns of this type are learned incrementally and forgotten in the same manner.

The second bird's responses (figures 5.4 a,b,c) did not have any really dominantly correlated propositions on the first day, other than 1,3, **right across a row** and **down a column**. The correlation amplitudes shown are of the order of 0.6 or less. The 2,7 and 2,8 propositions, which are **left along a row** and **down a positive slope**, **left along a row** and **up a positive slope**, were also employed to some extent.

The use of the row-column propositional combinations appear to suffice for this individual. There was no indication that the so called 'good' search was implemented in a directed manner. This bird did, however, sample more frequently than its predecessor and attain approximately the

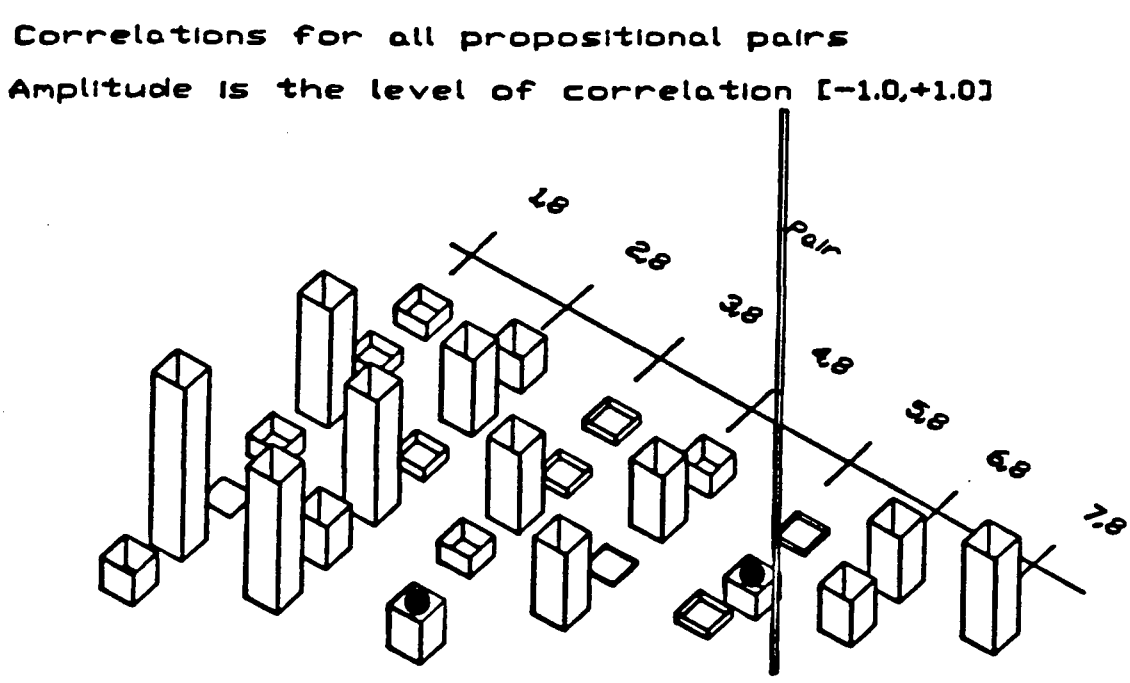


figure 5.3(a)

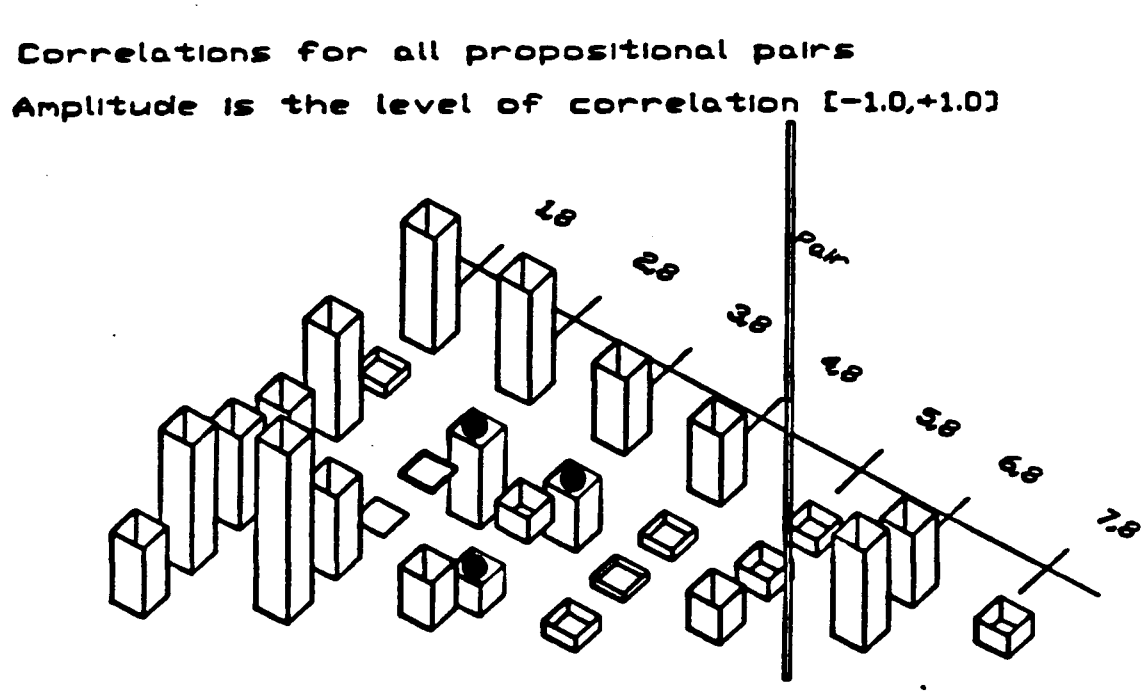


figure 5.3(b)

figure 5.3(a,b)

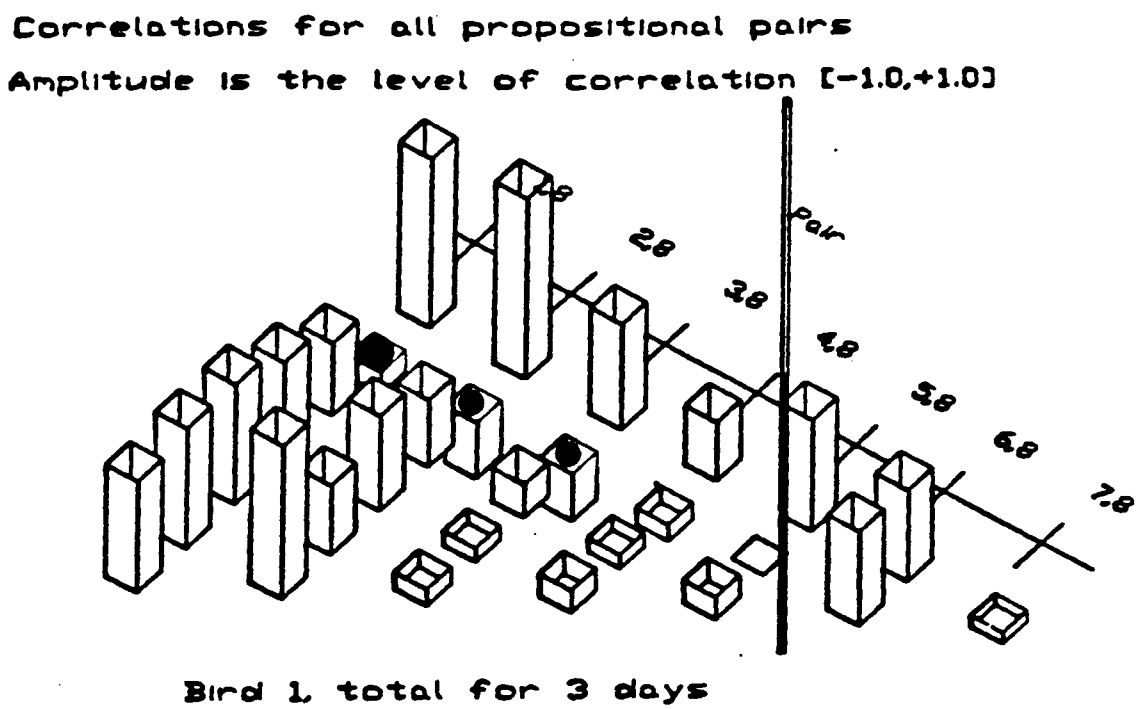

Correlations for all propositional pairs
Amplitude is the level of correlation [-1.0,+1.0]
Pair
Bird 1, total for 3 days

figure 5.3(c)

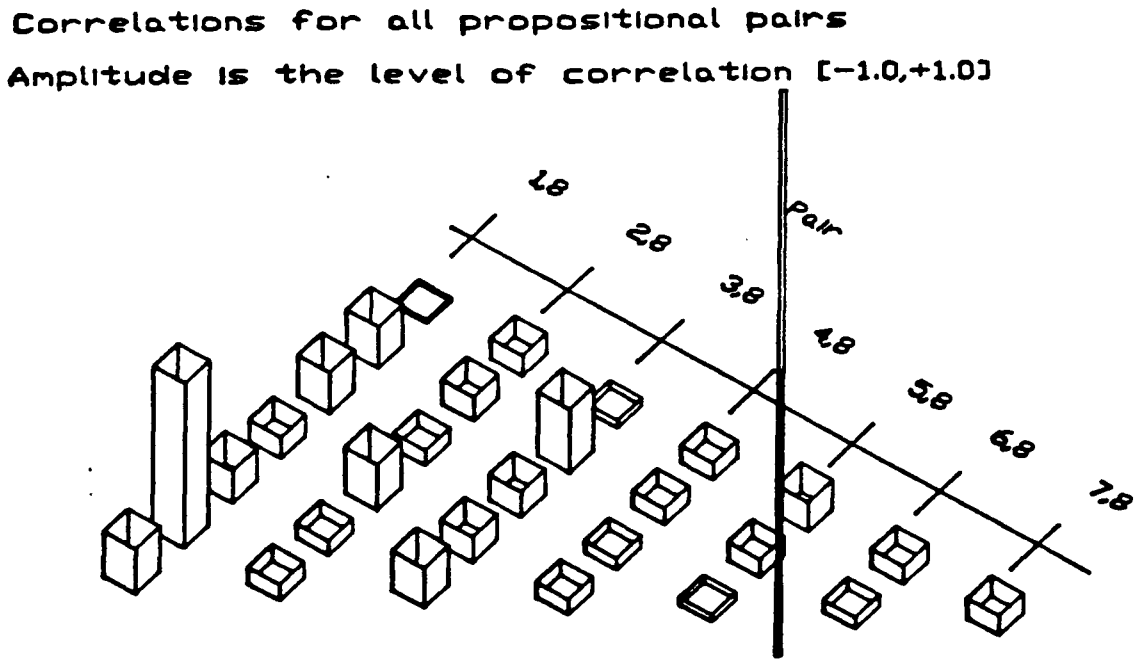


figure 5.4(a)

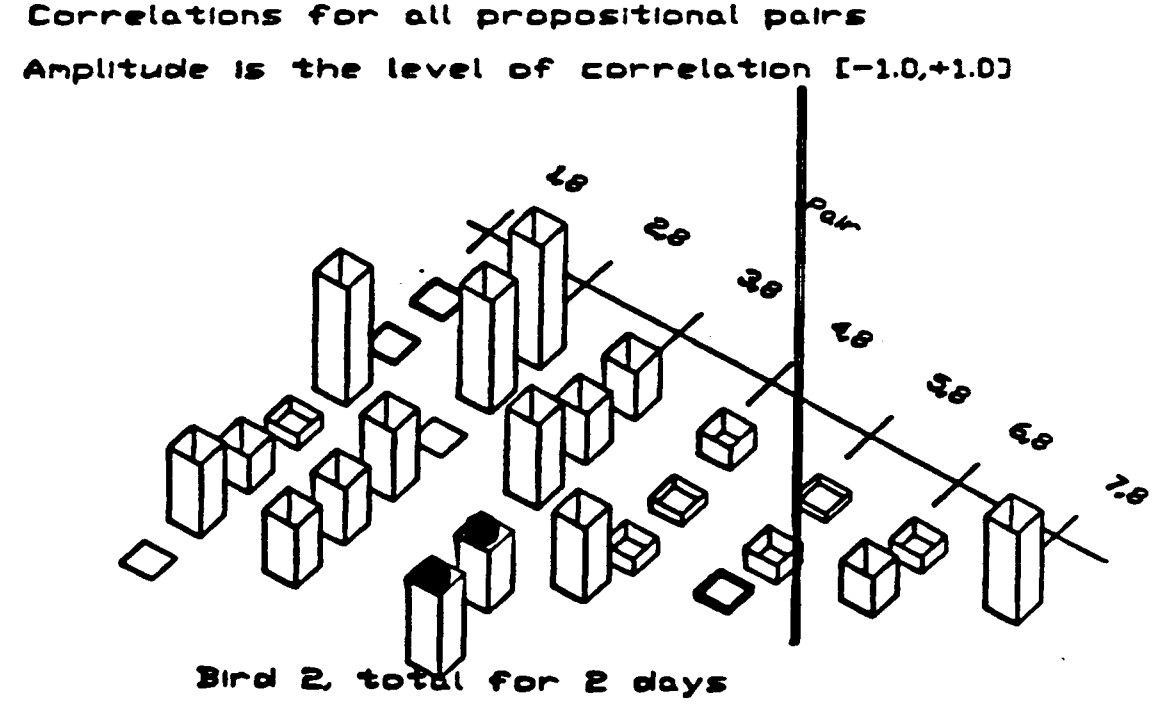


figure 5.4(b)

figure 5.4(a,b)

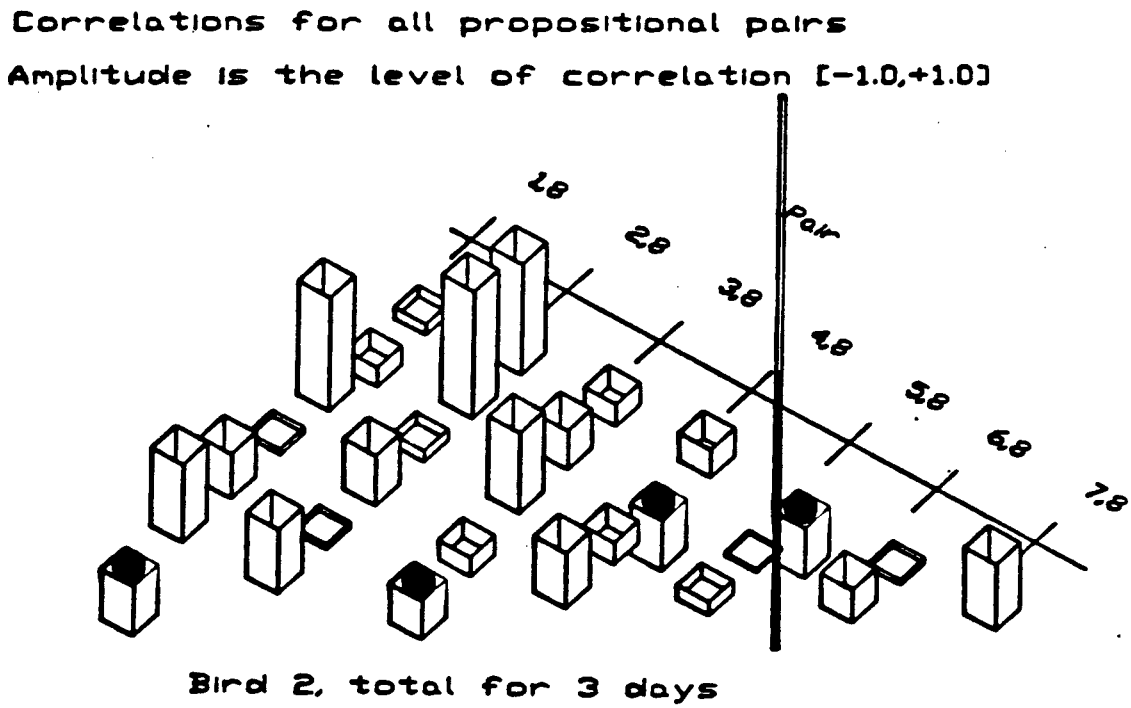

Correlations for all propositional pairs
Amplitude is the level of correlation [-1.0,+1.0]
1.0
2.0
3.0
4.0
5.0
6.0
7.0
pair
Bird 2, total for 3 days

figure 5.4(c)

same level of overall success. This seems to indicate that sampling more frequently will make up for a lack of recognition of the design.

The third bird (figures 5.5 a,b,c) was the most diverse in directly selecting propositional pairs as information carriers. The first day has the right and left row propositions used in conjunction with column searching. The second day has a diversification into the slope dominant propositions. The figure eight cycling was also observed with the slope propositions dominating. The correlations here are high, around 0.8, indicating that there was a high degree of coordination of these search types.

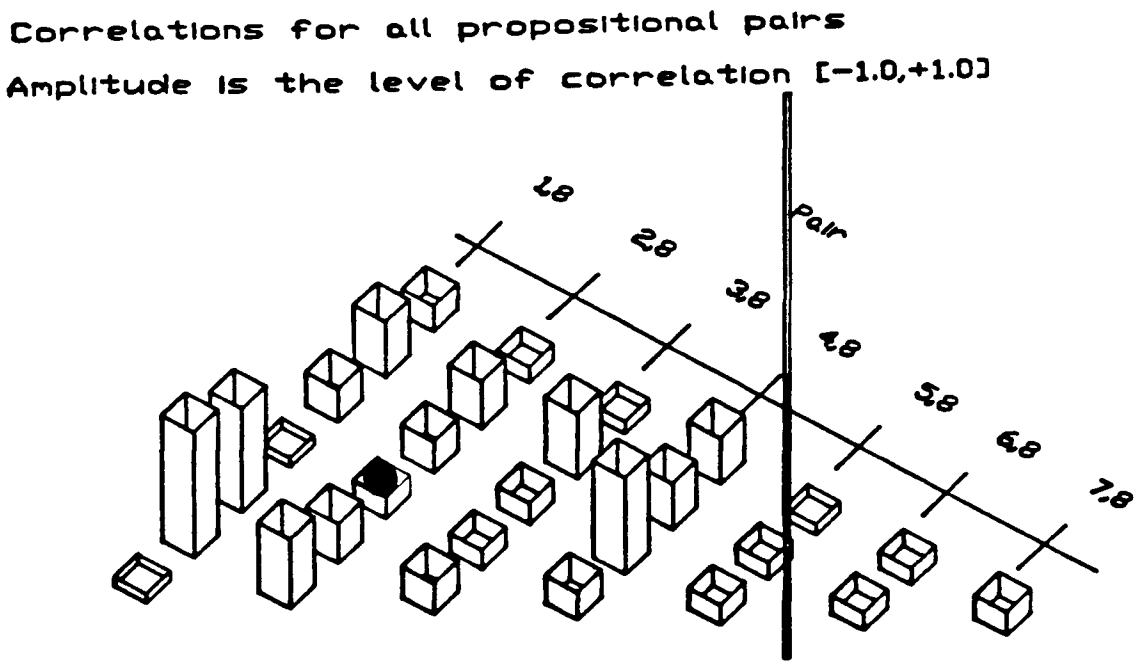
The fourth bird's responses (figures 5.6 a,b,c) are inconclusive. It would appear from the correlation amplitudes that 2,1 through 2,5 were selected, among others, since they are the only apparent peaks. However, the number of recorded responses per trial sometimes numbered as few as 4, making the selection of a rule from a maximal subset an arbitrary affair. The results are presented only for completeness and to illustrate a case where the analysis breaks down.

The fifth bird selected propositions for row-column search on the first day and the slope propositions (figures 5.7 a,b,c) on the second and third. Its responses are similar to those of the third bird but they appear to be selected in a different sequence. The figure-eight cycling is also observable. Here for the first time there are strong negative correlations indicating that some pairs of pair combinations were deliberately avoided.

5.4 Analysis of the Variation in Use of Propositional Pairs

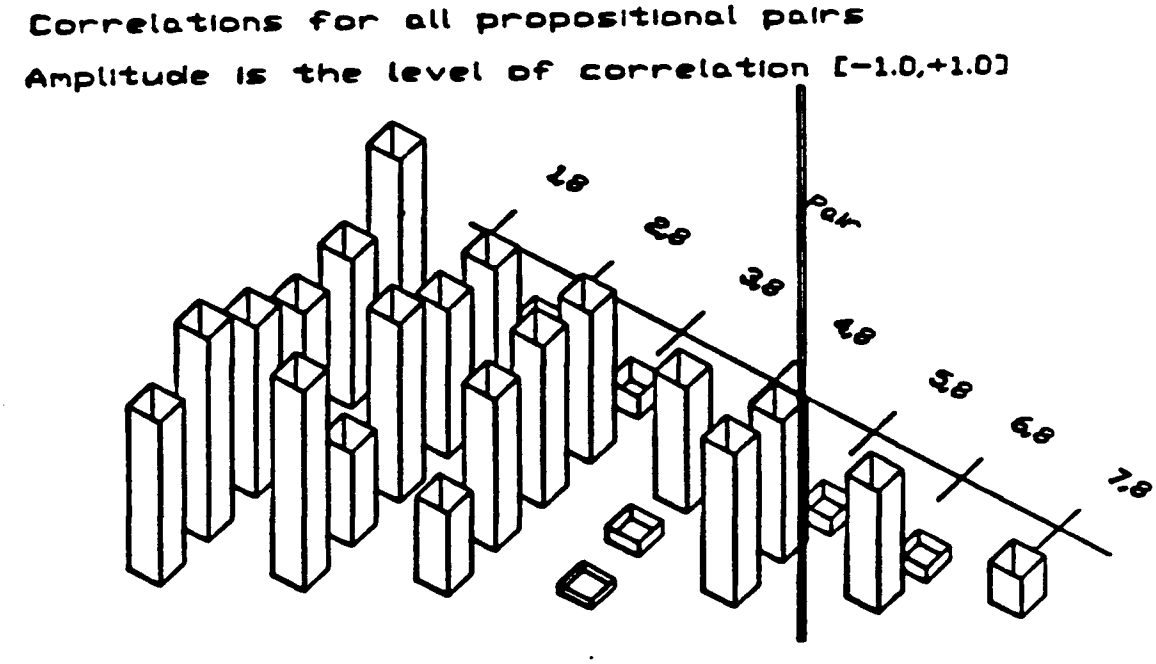
The previous section investigated correlation content in propositions. This section deals with fluctuations in the variation of selection of propositional pairs across a day's sampling. Cycling of propositional pair use was observed when the maximal subset algorithm worked its way through the data bases. It was decided that there was sufficient periodicity in this behaviour to try an analysis of the frequency in the use of selected propositional pairs.

The third day's foraging was chosen to illustrate the cycling in rule selection. The variance for a proposition pair was computed for the first 20 trials and the last 20 trials and compared to see if the sequences demonstrated stationarity. The difference was typically in the order of 15%. Considering the fact that the design was switched during the 25th trial this is not a large fluctuation.



Bird 3, total 1 day

figure 5.5(a)



Bird 3, total for 2 days

figure 5.5(b)

figure 5.5(a,b)

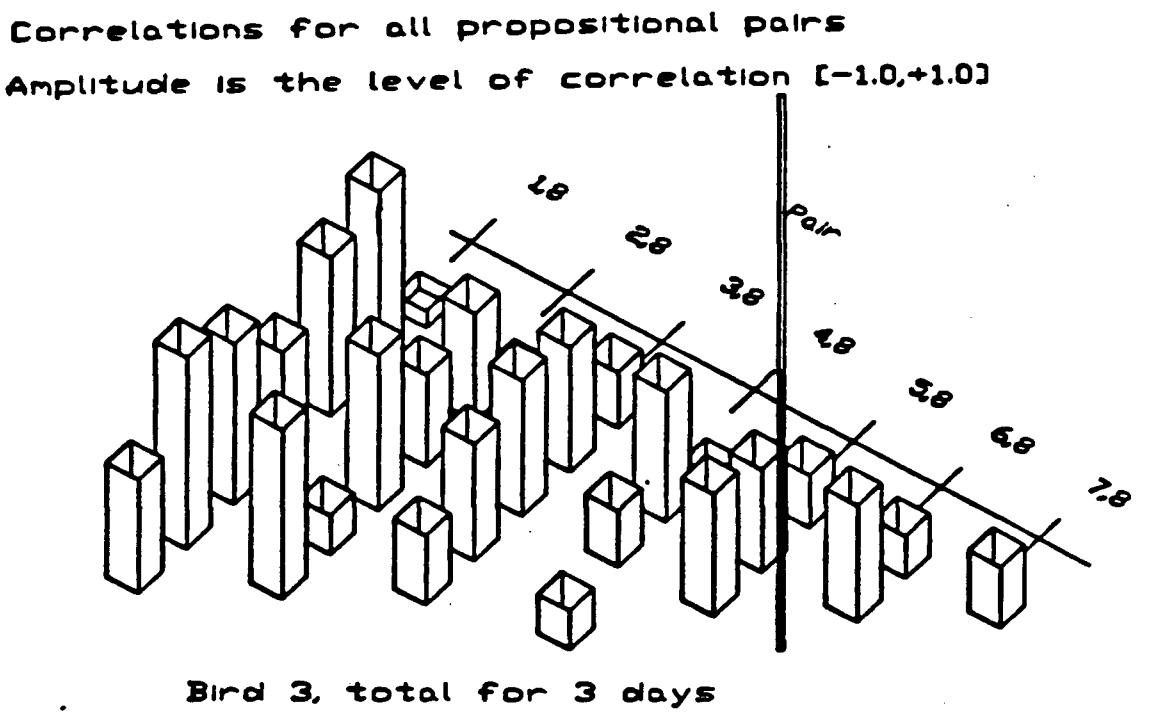

Correlations for all propositional pairs
Amplitude is the level of correlation [-1.0,+1.0]
1.0
2.0
3.0
4.0
5.0
6.0
7.0
pair
Bird 3, total for 3 days

figure 5.5(c)

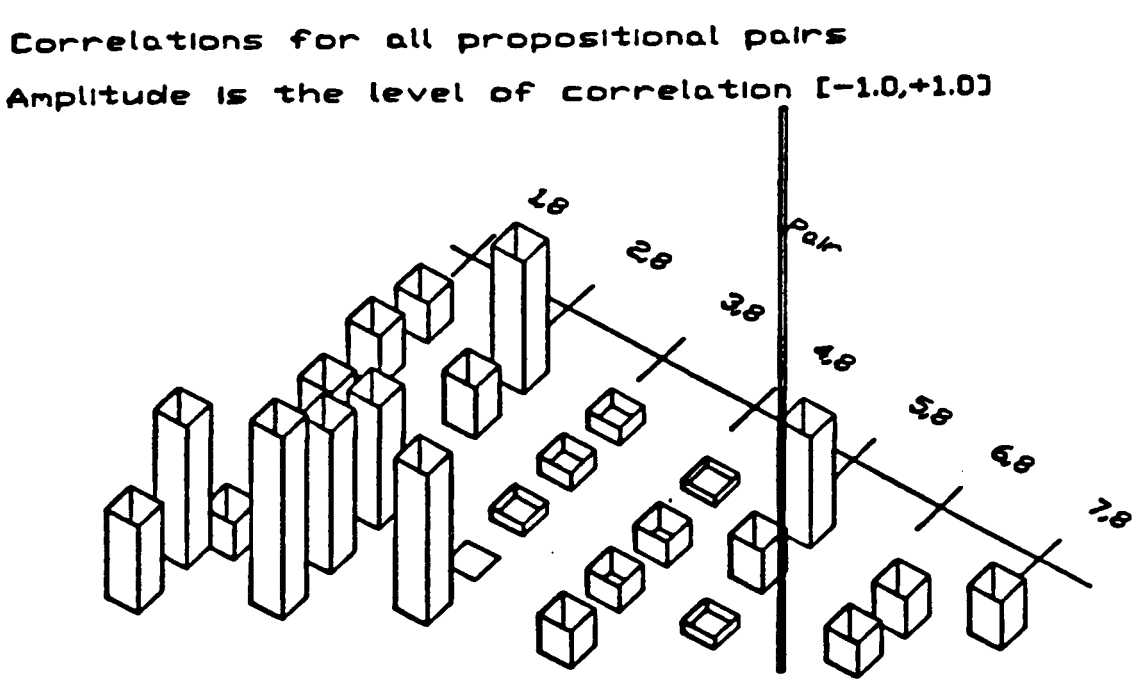


figure 5.6(a)

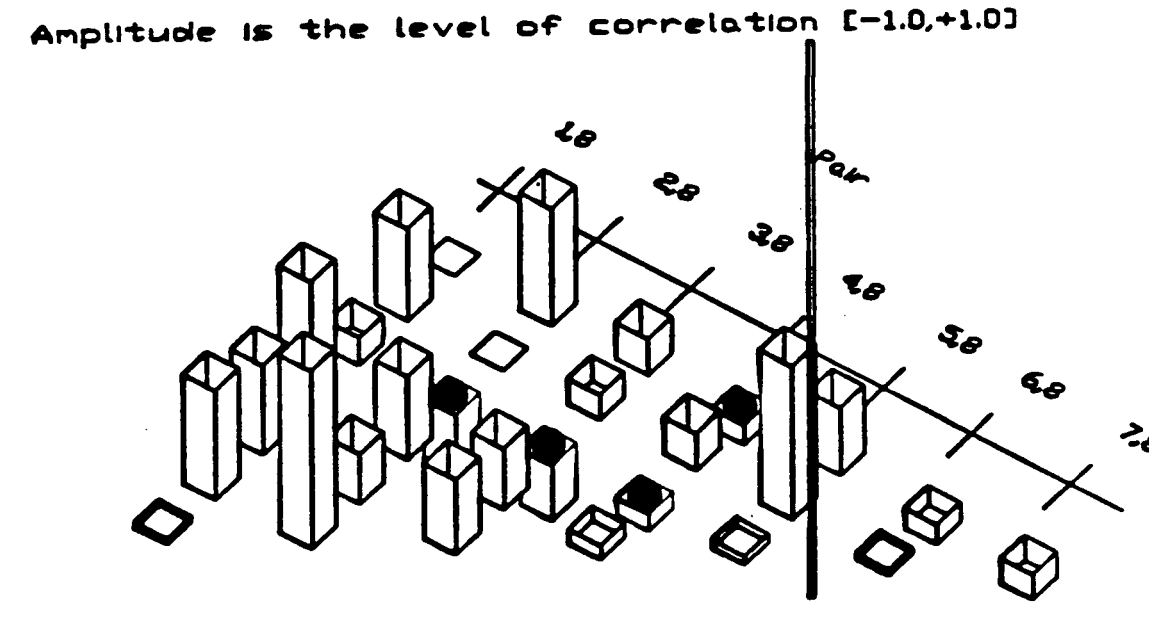


figure 5.6(b)

figure 5.6(a,b)

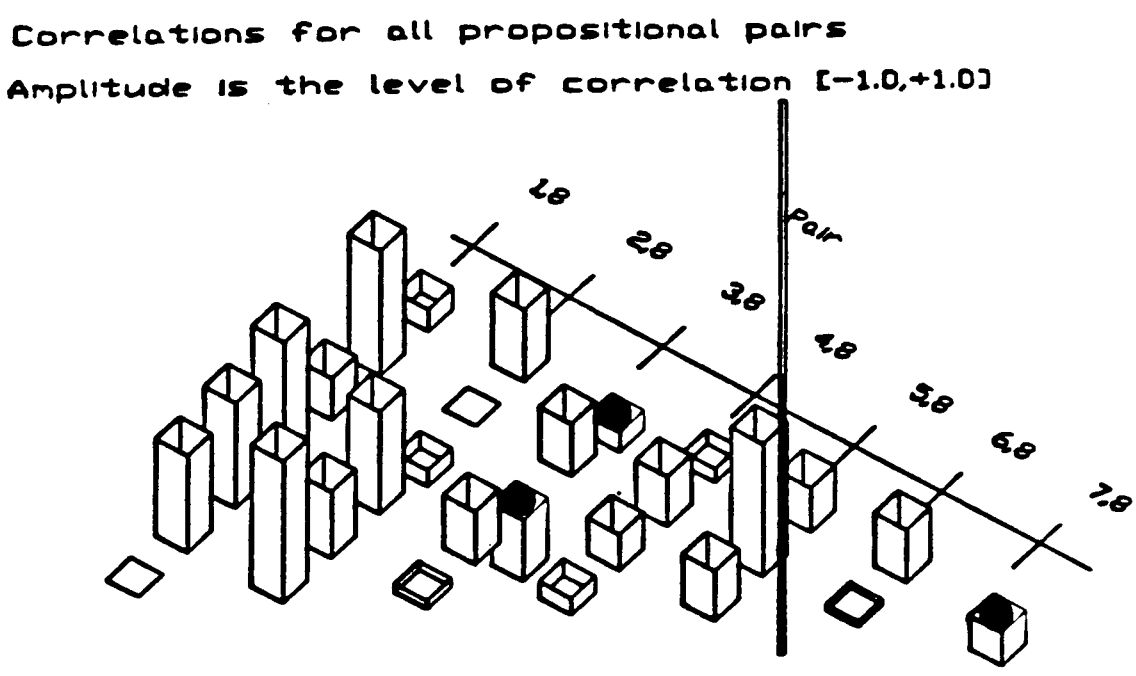

Correlations for all propositional pairs
Amplitude is the level of correlation [-1.0,+1.0]
1.0
2.0
3.0
4.0
5.0
6.0
7.0
Pair
Bird 4, total for 3 days

figure 5.6(c)

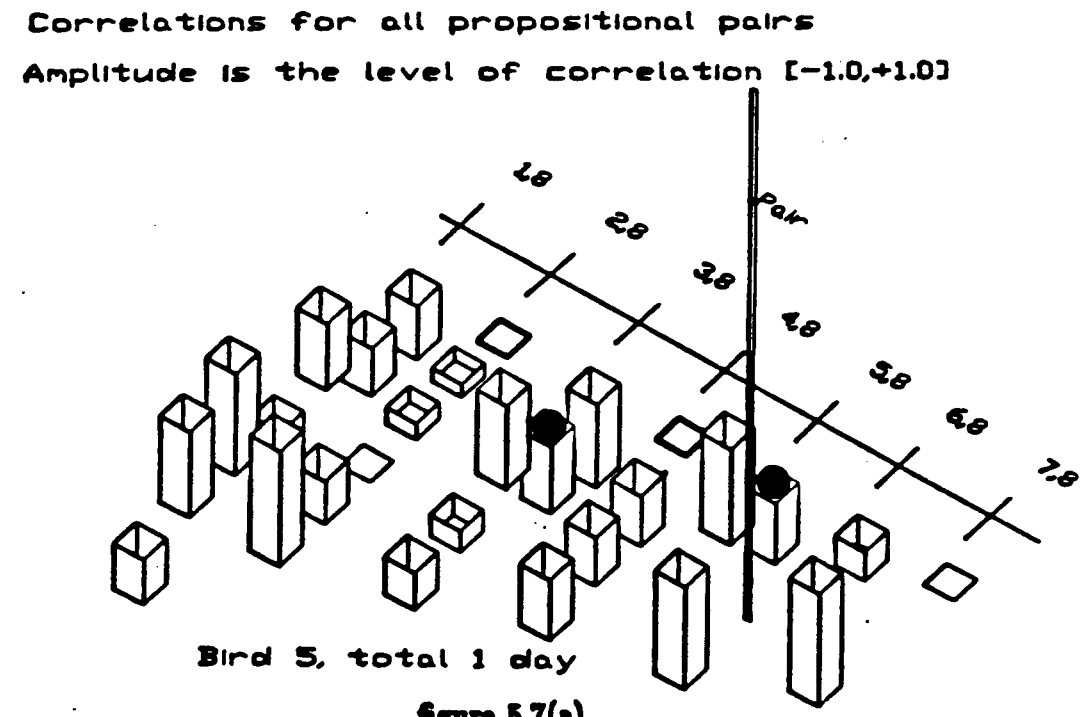


figure 5.7(a)

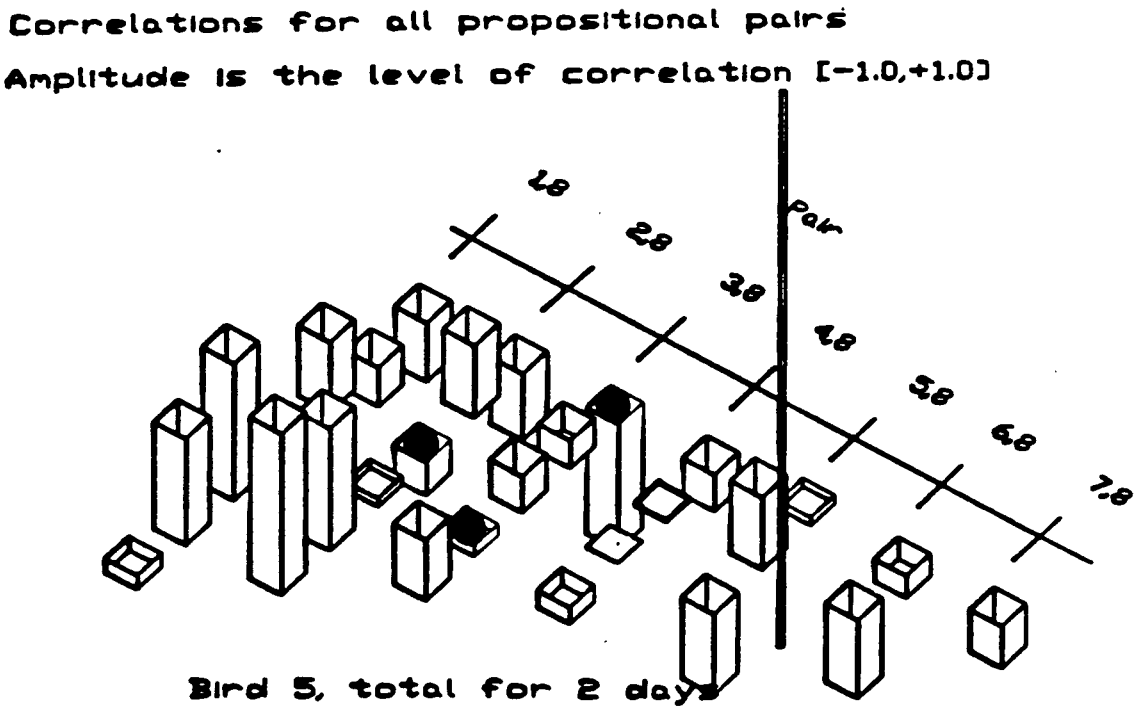


figure 5.7(b)

figure 5.7(a,b)

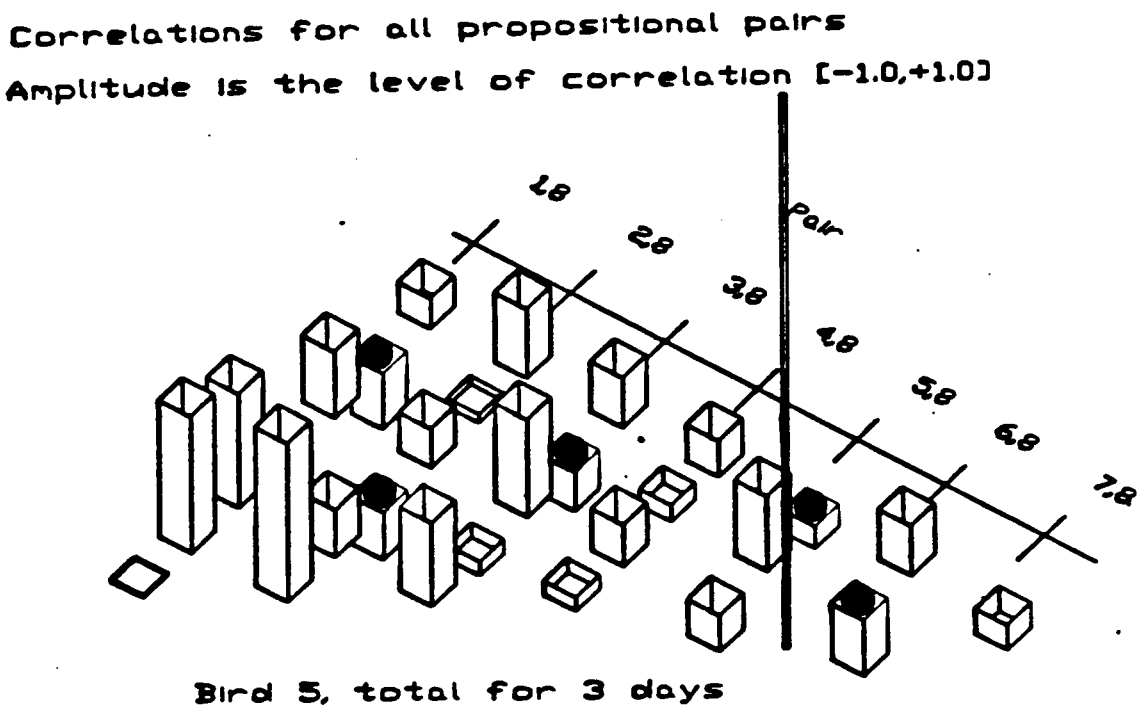

Correlations for all propositional pairs
Amplitude is the level of correlation [-1.0,+1.0]
1B
2B
3B
4B
5B
6B
7B
pair
Bird 5, total for 3 days

figure 5.7(c)

The time series was constructed as shown in figure 5.8.

The total number of times a proposition was detected during each 8 s interval in a 40 s trial was written out to a file. This procedure was repeated for each of the 40 trials and the result was the generation of 200 amplitude observations for each proposition. The assumption here is that even though there was approximately 10 min between trials, this factor would not greatly alter the use or non use of a particular proposition pair. From viewing the animation of the rule selections this assumption appears to be generally true. The sampling interval for the series is, therefore, 5/trial.

Each of these time series had the mean and linear trend removed. Each series pair has the lagged covariance C_s computed as shown in formula 5.1

$$C_s = \Sigma_{t=0}^{m} X_t \, Y_{t-s} \qquad 5.1$$

where X_t is the first time series, Y_{t-s} the second at time lag s. These were then multiplied and summed over the range m of the time series. Finally, the discrete Fourier transform was computed using the result of 5.1. The resulting squared amplitude spectrum represents the squared modulus of the transformed sequence. The transformation was computed using a FFT(fast Fourier transform) algorithm that implemented formula 5.2.

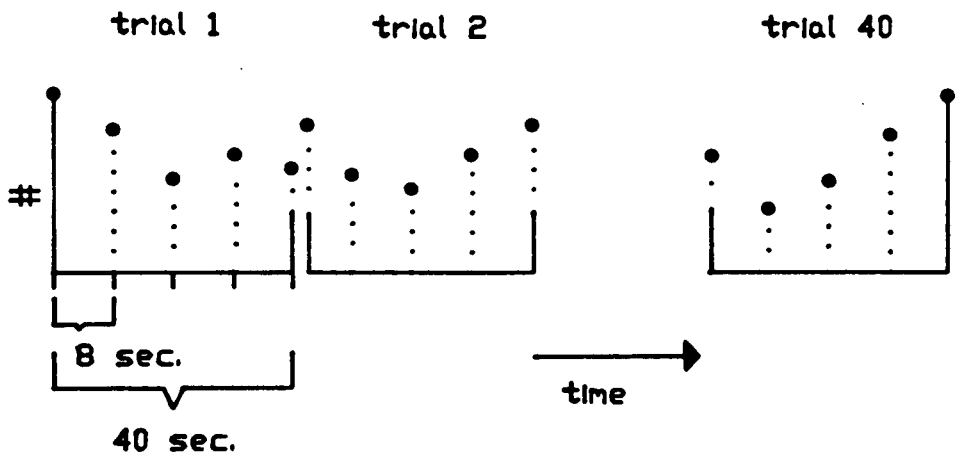
$$\Sigma_{s=0}^{nlags} |C_s z|^2 \qquad 5.2$$

where $z = e^{2\pi i f s}$, C_s is the covariance at lag s from 5.1, $nlags$ were the number of lags at which C_s was evaluated and f are the frequencies of the transform. $|.|^2$ indicates the square modulus of the transform. Selected pairs are presented in figures 5.8(a through g).

The spectra for birds 1, 2, 3 and 5, for the combination **up a column** and **down a positive slope** illustrate that during the course of a day's foraging each individual utilized the same rule combinations very differently. This can be seen by the location of the peaks in variation. Since these spectral profiles appear to be unique for each individual, future experimental designs could be made to encompass a spectral stereotype that is typical of an individual. These then would be incorporated in the expanded rule base as aids to instantiating a different type of pattern.

The fact that there are very strong peaks for particular pair selections across the entire sampling band raises the question of whether the birds have some kind of detection threshold that will allow

An illustration of how the time series were generated



= number of times a rule pair was used in an 8 sec. interval

figure 5.8

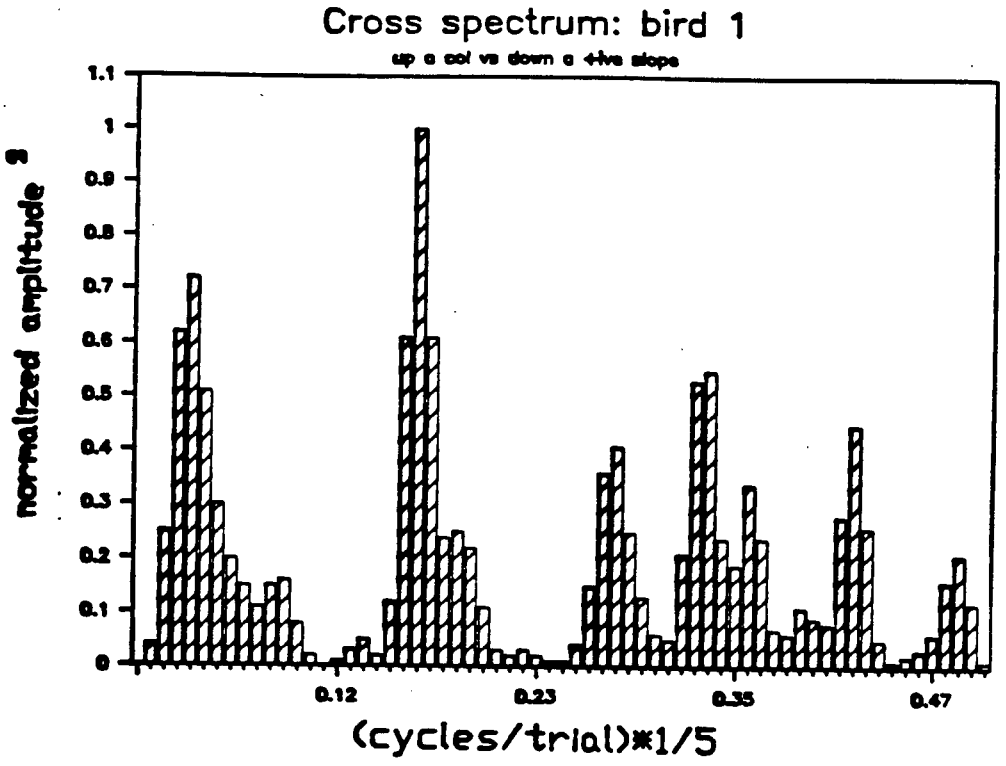


figure 5.8(a)

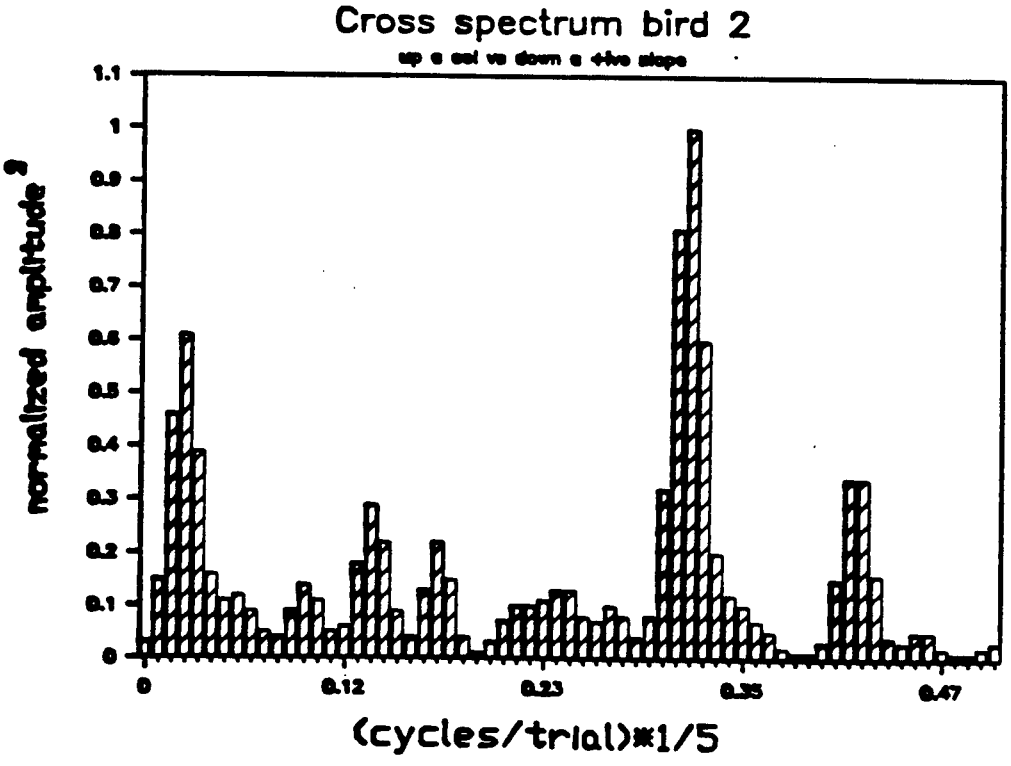


figure 5.8(b)

figure 5.8(a,b)

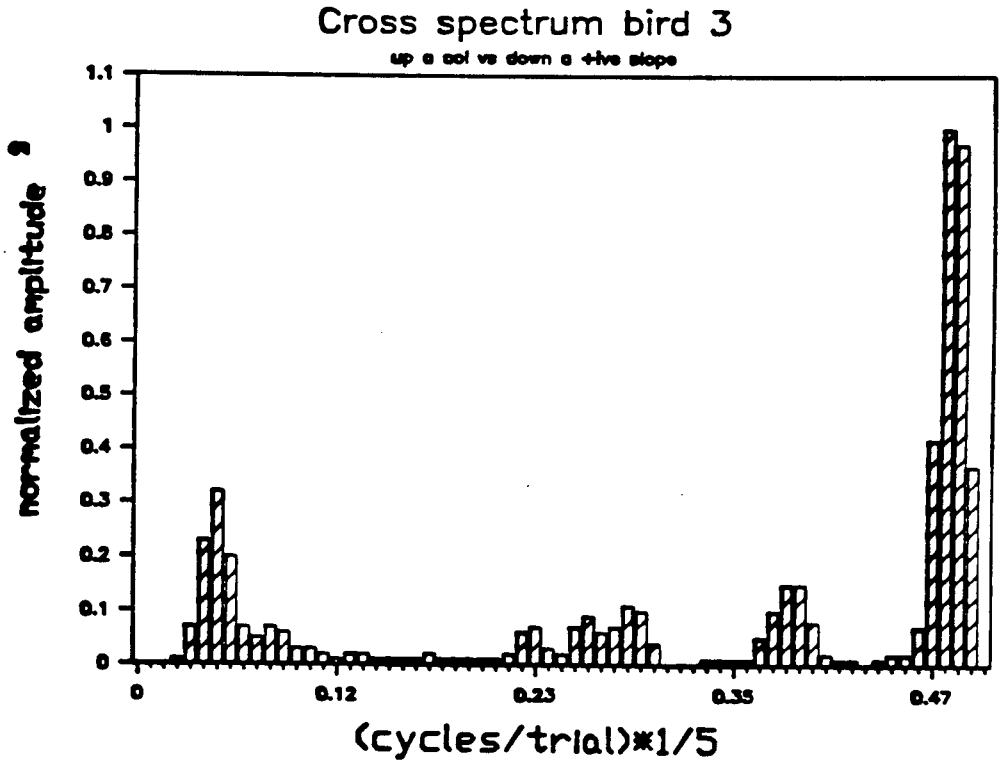


figure 5.8(c)

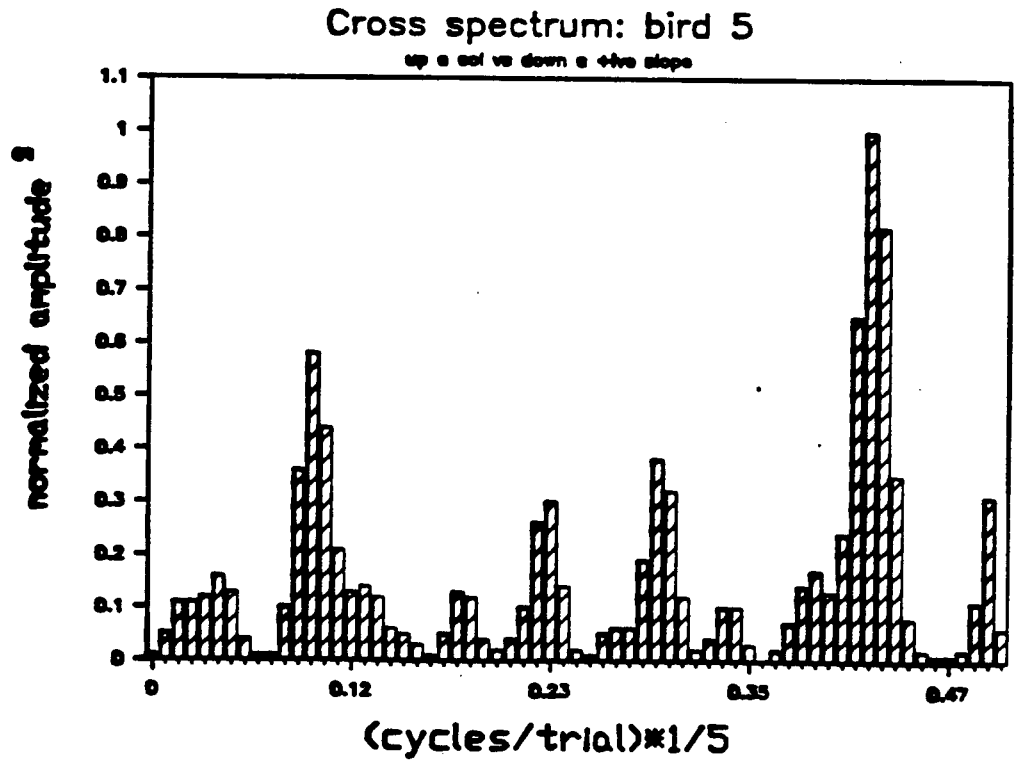


figure 5.8(d)

figure 5.8(c,d)

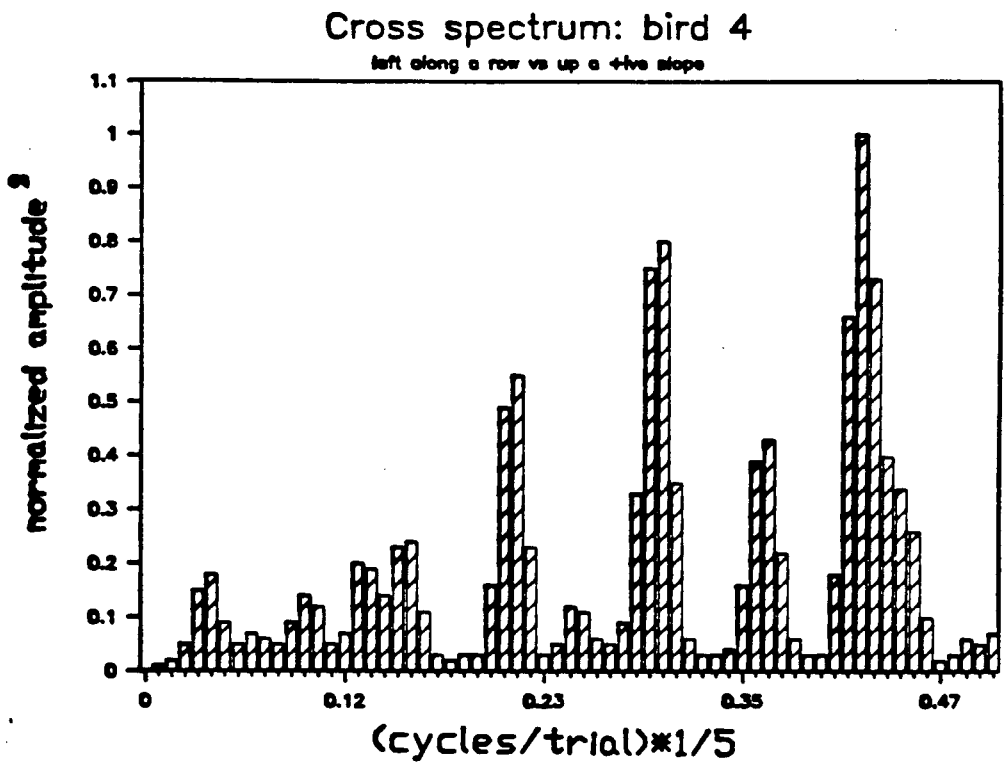


figure 5.8(e)

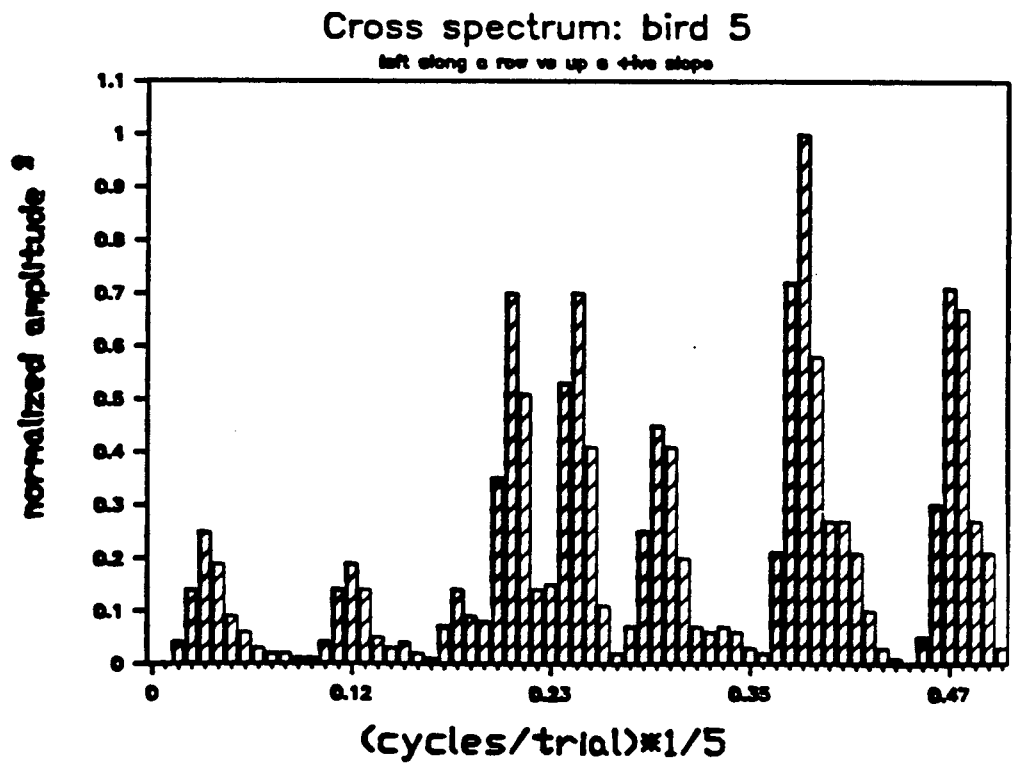


figure 5.8(f)

figure 5.8(e,f)

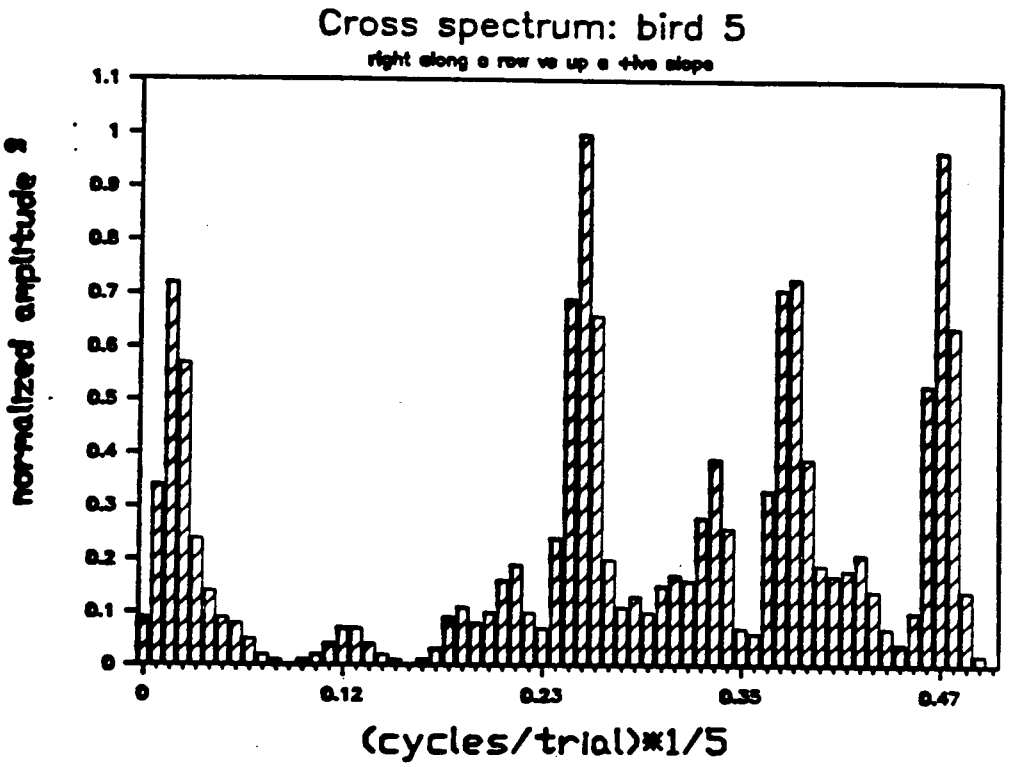

Cross spectrum: bird 5
right along a row we up a +ive slope
normalized amplitude S
1.1
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.12
0.23
0.35
0.47
(cycles/trial)*1/5

figure 5.8(g)

them to let go of a particular pair and take up another. It is not possible, in the context of this data base to establish this result with any certainty. However, there seemed to be a systematic transition from the simpler row-column combination, to those utilizing the diagonal structure of the design. This was most apparent during the third day's foraging.

5.5 Warping of Trajectories in One-Dimension onto a Target Trajectory

The one-dimensional warping algorithm developed in chapter 3.6, and appendix 2.1, was very useful in finding out which birds came closest to matching a targeted design. The so-called 'good search' trajectory outlined in Chapter 4 was coded up as the target and numerous blocks of foraging trajectories were warped onto it. The trials were blocked over a 5 trial interval. Intervals less than this level yielded little or no conclusive structure when warped. However, at this 5 trial level of aggregation it was possible to warp the behaviour onto the 'good search' target. The results for birds 1, 3, and 5 on the third day are presented in figures 5.9(a,b,c). Each line represents the sum total of frequency of visitation to a feeder location over trials 1 to 5, 6 to 10, 11 to 15, and 16 to 20 respectively. The warping parameter α was set to 0.15. A straight line at 45 degrees was included for a reference point. If the bird's trajectory matched the 'good' trajectory, then all of the points would lie along this line. As can be seen in the figures, the trajectories match well in the upper half of the grid. The first bird appears to have searched the nodes from 20 to 37 identically over the span of these trials. Things begin to unravel when one examines the matches in the lower half and the upper half of the grid. If a figure-eight loop was included as part of the 'good' design, then the warpings would be more exact and the match would be closer to the diagonal. It was these results that led to the conclusion that at least some of the spikes observed in the spectral ranges were due to the irregularities generated by this behaviour.

The trajectories were not warped over time. As can be seen in the spectral representations the ways in which these individuals went about learning patterns temporally was highly individualized. When selected pairs were warped onto one another, there were large deviations from the optimal line. They did not exhibit the regular behaviour depicted in figures 5.9(a,b,c).

The spatial trajectory model from Chapter 3 did not function well either. This model, that relied on probabilistic movements, did not come close to the patterning visible in the animations.

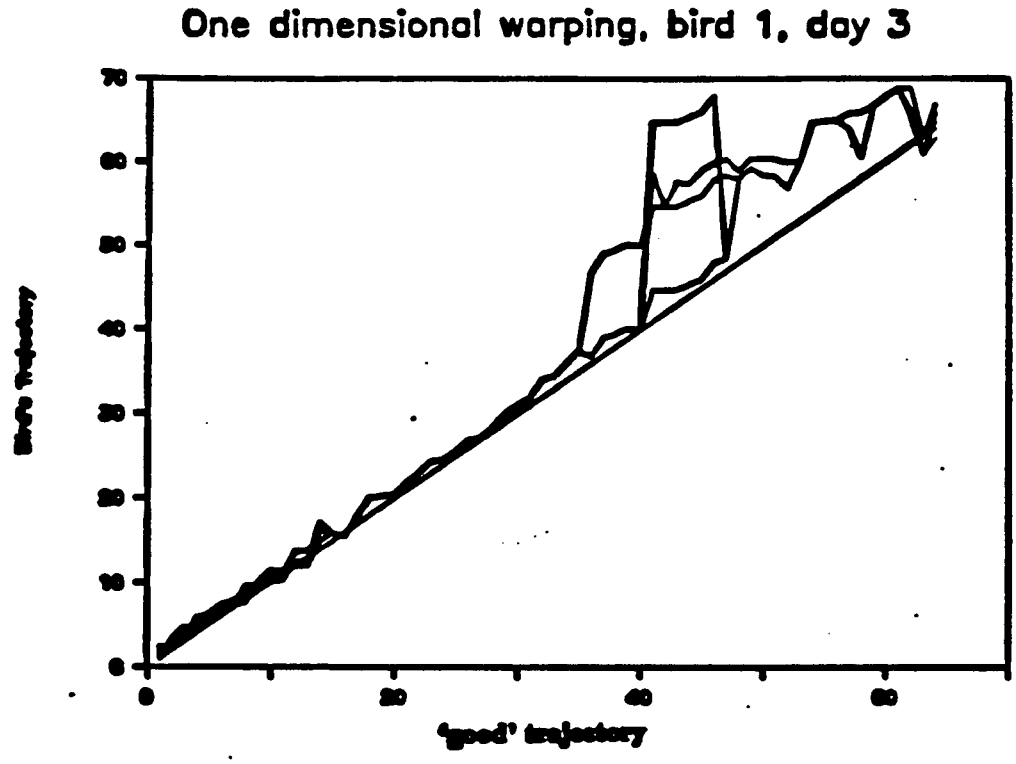


figure 5.9(a)

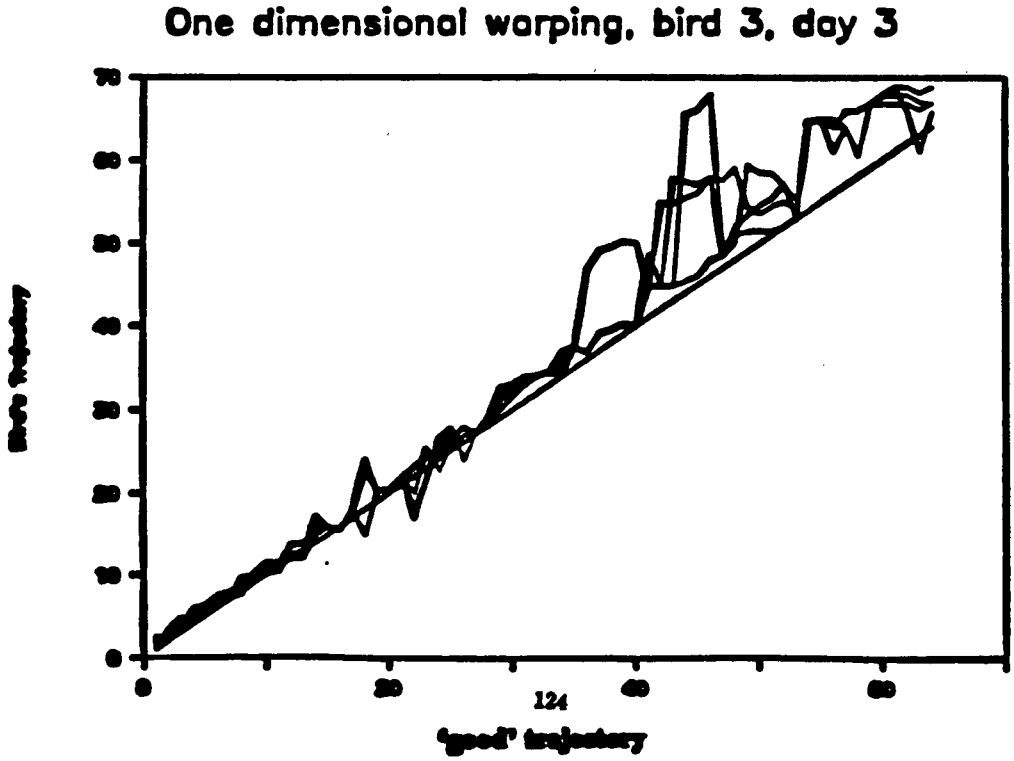


figure 5.9(b)

figure 5.9(a,b)

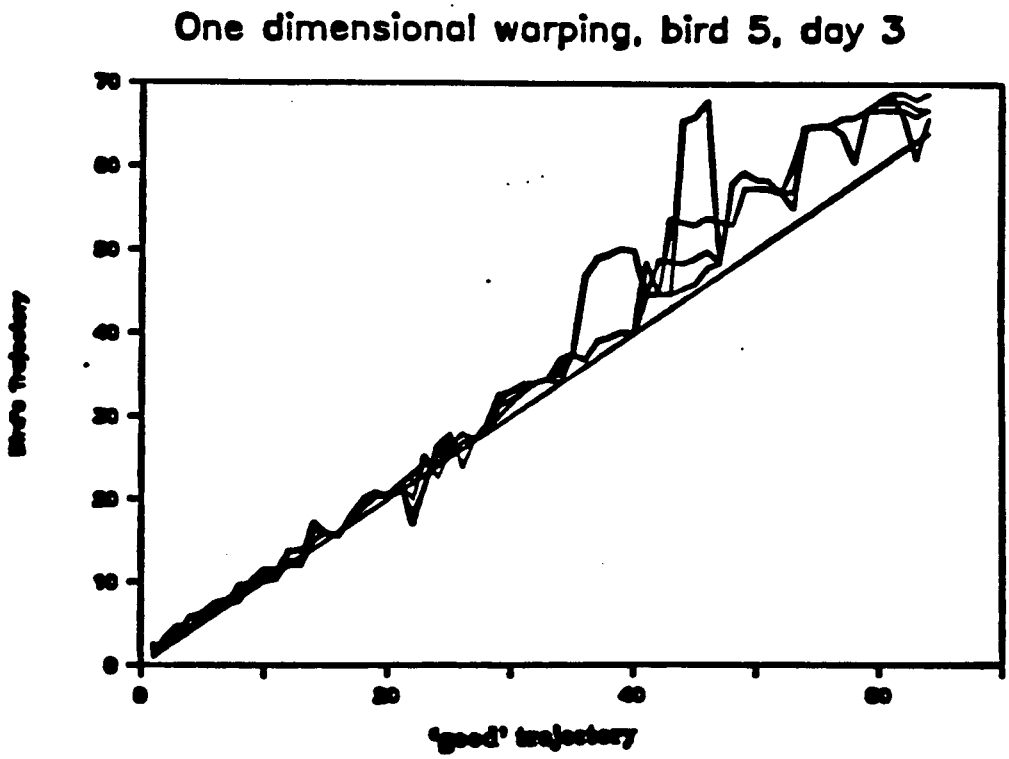


figure 5.9(c)

It was possible to achieve a roughly 60 % success rate with the model, but little was learned from using it by itself. Future research could use some type of probabilistic model, but it would have to be incorporated within the pattern recognition methods developed in this thesis.

5.6 Summary of Chapter 5

- A propositional lattice was developed to group rule type sequences.

- Cross correlations were computed for selected pairings of rule frequency sequences. The corresponding spectra were then computed using the Fourier transform.

- Simple correlations were computed for all rule frequency sequence combinations, for each bird, for 3, 2 and 1 days respectively. These were then presented as 3D histograms.

- The bird's search trajectories were warped onto a sample 'good' trajectory and the results compared.

Chapter 6

6.1 Summary and Conclusions

This thesis established a systematic way of representing a knowledge base for examining hummingbird foraging behaviour in a laboratory environment. It has succeeded in importing an array of customized symbolic and numeric techniques that have been developed in other disciplines and applying them to a complex heuristic problem. Using simple examples, it was shown that hummingbird foraging behaviour can be used in different and subtle ways to test the validity of the coded knowledge base. One example was the discovery that the figure-eight cycling that was observed in the animations could be coded simply as a combination of two similar diagonal search rules. This interactive approach also highlighted gaps in what is available as codable knowledge, and what must be learned about the structure of spatial learning mechanisms. For example, there were no instances of rules being used contiguously in groups of three at a time. The largest sequence found was two; even though the capacity to detect larger groupings was present, it was never used. The insights that have been gained from a systematic application and revision of the knowledge base are useful for modifying the designs and goals of future experiments. For example, the detection of periodic fluctuations in which pairs of rules were currently favoured over others indicates that a design that varied periodically over time could investigate this aspect of spatial learning.

As a descriptive tool for investigating how hummingbirds organize, remember and learn spatial patterns, this interactive approach has proved useful, illustrating that it is possible to compose a knowledge representation system from a small collection of simple facts and to apply it successfully to a complex problem. For example, by using only eight movement rules, most of the observed behaviour could be adequately investigated.

The techniques were especially useful in demonstrating that the same spatial pattern can be learned in five different ways by five different birds who enjoyed roughly the same level of success. Individual differences in ways of searching demonstrated the diversity in how the components of a spatial environment could go together to form a composite strategy. The organization, remembering, and learning of rule fragments is not a simple process of cataloguing and recall. The adaptive mechanisms demonstrated by the birds as the experiment unfolded were easily recognized using

the animation techniques outlined in Chapter 4. These learning strategies are simple enough for the human eye and brain to pick out, but very difficut to encode into simple rules in a knowledge base. The visual recognition step is a formal part of this integrated system, and has proved to be indispensible in formulating model templates. A system such as the one developed in this thesis provides a flexible environment for developing further experimental designs that could address the observed strategies on an individual basis. This system would allow closer investigation of how these birds put a knowledge base to work on other spatial designs. The idea of using a rule base to test a knowledge representation system is applicable to a wide variety of other systems.

A recent paper by Gould and Marler, (1987), illustrates that other organisms share the hummingbird's strategy, as observed by Sutherland,(1986). These animals first obtained an overall spatial pattern and then proceeded to refine it to specific instances. Gould and Marler showed that honey bees instinctively recognize different types of flowers as generalized images, and not simply by a list of characteristics. Gould then proceeded to list the hierarchy of elements that compose "bee knowledge". The same authors describe the application of a similar hierarchy by blackbirds learning to avoid, or to attack predators around their nests. These kinds of hierarchial representations of knowledge are beginning to emerge as ways that can be utilized to investigate how organisms organize their environments spatially. Gould and Marler conclude that

"much of learning, even though it is based on conditioning, is specialized for the learning of tasks the animal is likely to encounter. The animal is innately equipped to recognize when it should learn, what cues it should attend to, how to store the new information and how to refer to it in the future."

From the simple prototype of hummingbird foraging it is clear that the amount of information that can be gleaned visually and conceptually from pattern matching techniques greatly surpasses that yielded by a strictly numerical approach.For example, the use of animation to enumerate different search types simplified the range of patterns to be coded into the data bases. The combination of numeric and symbolic techniques permits many different points of view for a particular problem. The viewpoints can be changed quickly to suit many different spatial and temporal scales in an ongoing analysis. The drawback of symbolic processing is its combinatoric nature. The searching complexity quickly overloads the computing resources available to a micro-computer.

The reasoning mechanisms developed for this particular application have proved to be very general. The maximal subset finder, theorem prover, instance lister, uncertainty reasoner, and language parser are all currently used as teaching aids for an extension course offered by the U.B.C. computing center. This course presents Arity prolog, a dialect of prolog that is used on micro-computers, as a development environment for expert systems. The course lecturer, Dr. Glen Cooper, has in most cases greatly extended both the flexibility of the original algorithms and their features in order to illustrate the inner workings of first order predicate logic. The prolog programs are stored in the MTS computer ID "AI". All of the file handling utilities, parsers, etc., are now contained in a file named toolbox.ari.

6.2 Concurrency and Spatial Learning

Concurrency of events is so common in natural processes that it is generally taken for granted and overlooked by experimenters. The concurrent environment in which these hummingbirds exercise their spatial learning abilities is in direct conflict with the sequential nature of the digital computer. In order to model some of the components of the spatial learning process, the experimenter attempts to mimic these concurrencies with a list of sequential intructions. This is not a problem if the learning is viewed as simply occurring sequentially in time, but it is clear from the illustrations in this thesis that hummingbirds and other animals do not use a simple sequential learning process. The birds learn concurrently on several different spatial and temporal scales. For example, the birds' initial understanding of the grids' spatial pattern mainly involved searching in halves or quadrants. As time progressed, smaller isolated groups of filled feeders were recognized and the gross level searches resulted in fewer errors. The first bird tested had a 17.3 hour waiting period between the first and second day's foraging. Somehow, during this time it had solved a sizable portion of the diagonal structure of the spatial pattern. When it began foraging on the second day the searching along the diagonals had much fewer errors than at the end of the first day. It would be preferable to have several process models working on several different learning scales and communicating their findings to one another, then continuing their individual searches. The simplest level of pseudo-concurrency is the swapping of tasks quickly in and out of computer memory, so that they appear concurrent. The whole process still has only one beginning and one end. Concurrent processes have many beginnings

and many endings. To begin the process of assembling all of the elements to investigate spatial learning in hummingbirds, a concurrent operating environment is of primary importance.

6.3 Changes Required in the Computing System Design

The structure of the system developed here was that of a prototype. A working system that would analyze spatial learning problems as they unfold would have to operate in a concurrent environment. This would require a real time clock, which was not present in the current system. The clock would be used to give an accurate temporal picture of the foraging, rather than this strictly spatial one. The UNIX implementation of a concurrent environment is not adequate for accommodating the graphics, serial communication, and numeric and symbolic analysis. The time slicing used by Unix quickly becomes ineffective when graphics or searching algorithms are working on the same data base. A system that could handle this computational load adequately would have to have a complete virtual MS-DOS emulation, a concurrent operating system that permitted graphics windowing, and most important, a parallel processing capability. Such a configuration is possible on higher-priced systems, such as the Hewlett Packard AI work station.

The grid used here included only 64 nodes, and thus severely limited the geometry of the spatial designs. Currently there are two other add-on boards being constructed for the hummingbird laboratory to handle input from 512 grid nodes through a serial port on a separate pc. Aside from the logistics of refilling the feeders at the end of each trial, there is the problem of processing the data. The code for this prototype was written, for various reasons, in four different languages: Prolog, C, Pascal and Fortran. Each of these languages deals with IO(input and output), and the operating system in different ways. The code, along with the data base management utilities, graphics, and BIOS(basic input and output system) control occupies some 27mb of storage space, which is distributed on two 20mb hard disks. In order for these various utilities to work together, a parallel processing environment, in the form of a transputer development board and operating environment would be required. The transputer is similar to the Von Neumann computer except that it resides on a chip and can be hardware linked, using software calls to other transputers forming a dedicated network that can process information in parallel. The IO for these transputers is around 10mips(million instructions per second). The language used for the communication with

this hardware is called Occam; it is the first truly parallel processing environment available for micros and main-frames. In order for the IBM AT system to function while handling arrays of 500 nodes, all of the numerical computations must be down-loaded to the board containing the transputers. The results from its computations would then be written onto the hard disks. Having hardware as fast as the transputer would cause the resulting output to be bottle-necked through the AT's BIOS. Currently, programs using MSDOS can access only 640kb of RAM. The system requirements for the enlarged grid would require at least 3mb of RAM. If graphics applications are to be used at the same time, approximately 10mb of RAM would be required. Access to that much RAM would be possible if a Unix-MSDOS compatable system could access the transputers in protected mode. This configuration would require some customized hardware and at least two levels of board interfacing: one interface to write to video memory and another to handle file IO.

Ultimately, one would wish for an operating system that could be easily customized by the researcher to accommodate the task at hand. The so-called 5-th generation computers whose operating systems are written in Prolog are an example of the type of system where reconfiguring how the system hardware responds to a particular situation can be accomplished while the system is running. The "Lisp" machines are another example. Both of these options are prohibitively expensive and generally inaccessible to the individual. There are some inexpensive do-it-yourself operating system toolboxes, written in C, that provide a cross-over from MSDOS to UNIX. These are, however, quite complex for the ordinary researcher to implement.

6.4 Knowledge Representation in Biological Systems

Knowledge representation is a rapidly expanding field in AI. Many fields have realized its use in quickly making all information available, but biology has not followed suit. Many biological problems lend themselves to this hierarchical retrieval and analysis. This thesis provides a clear demonstration that until one can adequately represent the knowledge known, the knowledge that is acquired through experimentation will answer few questions. The lack of systematic algorithmic inquiry is one of the principle failings in biological experimentation. This thesis, hopefully, has provided a prototype model for future use in the design and implementation of knowledge representation in biological systems.

Appendix 1

1.1.1 Warping of Data Sequences That Have Been Irregularly Sampled in Time and Space

1.1.2 One-Dimensional Time Warping

The problem of matching foraging trajectories that have been irregularly sampled in time was introduced in Chapter 3. The simplest way to match trajectories is by stretching the coordinates of one onto those of the target. The development of the algorithm that implements this action was developed by Roberts(1984). In the case of the hummingbird foraging, the trajectories are the paths followed by birds around the grid during a trial. The locations on the grid are shown in Chapter 3, figure 3.1. The amplitude at any location is the number of times during a trial that the bird visited that location. The task then is to try and match any one of these trajectories onto a modelled trajectory, or another foraging path. The matching is done by optimizing the distance a point must be moved against a penalty function under the control of the Dynamic Programming algorithm. The name given to this process is Dynamic Time Warping. Roberts begins by stating that "*Dynamic Time Warping is the use of the Dynamic Programming Algorithm of Bellman(1957), to find from given trajectories a and b the warping function w that minimizes a specified cost. T is used to represent the number of locations at which each trajectory is sampled. The warping function w is a monotonic integer mapping from (1...T) to (1...T) with 0 being used to denote an undefined value. It is useful to visualize w as a path contained in the two-dimensional array of locations (1...T; 1...T), with t being the horizontal parameter and w[t] the vertical parameter.*" Chapter 3, figure 3.2. "*When the path goes outside the array, w[t] is then defined and is given the symbolic value of 0. The tail of a path is the subset of the path from some t on to the end at T. The w is a warping function such that when w[t] is used to replace t it produces a warped time axis. Thus, a[w[t]] is a function of t that is a warped version of a[t].* "

Roberts developed the warping algorithm as follows: "*Warping is composed of three operations. The first,* **shifting**, *is produced when* $w[1.0] \neq 1.0$ *meaning that the whole trajectory has been shifted along the time axis. The second, a* **jump**, *is produced when* $w[t + 1.0] > w[t] + 1.0$ *and so the time axis has been stretched near the points t and t 1. The third form of operation is a* **flat** *spot where*

$w[t+1] = w[t]$. *A warping function is composed of a shift and any combination of jumps and flat spots.*

The cost function is a real valued function of two trajectories a and b and a warping w. The inverse warping function w^{-1} *is the function such that, wherever w is defined and strictly monotonic,* $w^{-1}[w[t]] = t$, *and wherever* w^{-1} *is defined and strictly monotonic,* $w[w^{-1}[t]] = t$."

1.1.3 Design of a Cost Function

The next stage of Roberts' algorithm is the design of a cost function.

"*The cost function that is to be minimized is a function of the trajectories a and b, and the warping function w. The cost function used in this application is based on the correlation Q between trajectory* $b[t]$ *and the warped wave* $a[w[t]]$. *A high degree of correlation produces a low value of the cost function, and conversely, a low correlation produces a high value of the cost function. There is also a penalty function P that is a function of w.* α *is the coefficient of the penalty that controls directly how expensive the penalty is. The cost function used by this algorithm is*

$$C(a,b,x,\alpha) = \begin{pmatrix} [Q(a,b,w) - \alpha P(w)]^{-1} & [Q(a,b,w) > \alpha P(w)] \\ +\infty & [(Q(a,b,w) < \alpha P(w)] \end{pmatrix}$$

The numerical solutions to the cost function must be so designed that it can be calculated for the Dynamic Programming Algorithm. A necessary condition for this is that the cost of any tail from some t on to the end T can be calculated knowing only the tail and not the rest of the path. This condition arises from the way in which the Dynamic Programming Algorithm works. It is desirable to be able to compute this cost in a small number of operations as it will be calculated many times. The minimization of the cost function was translated to be a maximization of the denominator of the ratio:

$$Q(a,b,w) - \alpha P(w)$$

This allows the contributions of the tails to be calculated from any t if the contributions to P and Q of the tail could be calculated. The cost function itself could not be calculated for each tail and so could not be used directly with the Dynamic Programming algorithm."

1.1.4 The Correlation Coefficient Q

Roberts then defines a correlation measure and a penalty function with constraints.

"*The simple correlation coefficient*

$$Q = \sum_{t=1}^{T} a[w[t]]b[t]$$

is summed over t and not w[t]. This function is not symmetric with respect to a and b because

$$\sum_{t=1}^{T} a[w[t]]b[t] \neq \sum_{t=1}^{t} b[w^{-1}[t]]a[t]$$

It is necessary that the cost function and the solutions be symmetric between a and b. Specifically, if warping a to b produced a cost C and a warping function w, the warping of b to a should produce the same cost C and the inverse of the warping function w^{-1}. The inclusion of the sub sum

$$\sum_{w[t]>s>w[t-1]} a[s]b[s]$$

makes the function symmetric by summing all of the positions in a jump, just as all of the positions in a flat spot are summed. This results in a modified correlation quotient

$$Q = \sum_{t=1}^{T} \left(a[w[t]]b[t] + \sum_{w[t]>s>w[t-1]} a[s]b[t] \right)$$

1.1.5 Constraints and the Penalty Function

The path w is constrained to be monotonically increasing, which is a sufficient condition for the existence of a unique inverse. As the functions range over discrete time rather than continuous time, discontinuities in the inverse are not a problem. This allows sections of the paths to have slope zero (flat spots).

Dynamic Time Warping has been used in speech recognition (Rabiner, et al., 1978; Myers and Rabiner, 1981; Brown and Rabiner, 1982). In speech applications, the start and end of the word had been identified before the Time Warping was applied. This allowed the endpoints of the warping function to be constrained to be (0,0) and (T,T). "

Roberts(1984) decided that a more elegant method of producing reasonable warping functions would be to include a penalty for discontinuities in the cost function.

"*The penalty function could be of the form:*

$$P(w) = \sum_{t-1}^{T} |w[t+1] - w[t] - 1|^{n}$$

The parameter n determines how severely large jumps are to be penalized relative to small jumps. As n approaches infinity, the penalty function becomes the constraint used in speech recognition. For n=1:

$$P(w) = \sum_{t-1}^{T} |w[t+1] - w[t] - 1|$$

This computationally very simple penalty function did not strongly penalize large jumps, and so the warping functions were often severely discontinuous. Increasing the penalty coefficient, - α, enough to prevent large jumps eliminated the small variations as well.

For n=2:

$$P(w) = \sum_{t-1}^{T} |w[t+1] - w[t] - 1|^{2}$$

This penalty function is not symmetric in that the flat spots are penalized linearly rather than proportionally to the square of their length. Using the identity:

$$x^2 = \sum_{r=1}^{x} 2r - 1$$

and the formula for the length of the flat spot from t on:

$$inf(s - t - 1 \text{ such that } s \in \{1, ..., T]; s > t; w[s] \neq w[t])$$

produced the symmetric penalty function

$$P = \sum_{t=1}^{T-1} p(w,t)$$

where

$$p(w,t) = \begin{pmatrix} (w[t+1] - w[t] - 1)^2 & w[t+1] > w[t] \\ 2 \times inf[s - t - 1 | s \in \{1, ..., T\};\ w[s] \neq w[t]] - 1 & w[t+1] = w[t] \end{pmatrix}$$

This function solved the problem of severe discontinuities by penalizing large jumps or long flat sections very severely. A disadvantage of the squared penalty, that all such nonlinear penalties share, is that the dynamic programming algorithm may not produce the optimal solution. The penalty for a section with slope zero is the square of the length of the section. The previous length of the flat spot affects the penalty for continuing the flat spot. This means that locally optimal paths may not be globally optimal, i.e., the best path from (s,j) to (t,k) may not be a subsection of the best path from (r,i) through (s,j) to (t,k). This may result in paths that, when traced backwards, have a zero slope section starting one place too soon. This is not a major problem as the algorithm is robust and the paths found will be very close to optimal."

The coefficient α of the penalty function controls directly how expensive it is to warp. A small (0.01) value of α produces extreme warpings. A large (0.2) value of α produced warping functions that were merely shifts with neither jumps nor flat spots. By trial and error a value of (0.13) was proven to be the best compromise between resolution and accuracy.

1.1.6 Summary of Appendix 1.1

- The problem of analyzing irregular time varying time sequences was presented.

- The dynamic time warping algorithm was presented along with its application to a laboratory foraging sequence.

1.2.1 Two Dimensional Spatial Warping

Although the grid on which the hummingbird foraged is uniform in design, the way in which the bird sampled the different sections was highly non-uniform. The amplitude measurement at each of the nodes was the total number of times this node was visited during a 40 second trial. The non-uniformity in the sampling was assumed to be due in part to errors in the bird's perception of where the boundaries of the experimental design lie, and which sections appear to contain the most reward. Continuing with this hypothesis requires that the features of this sampling space be simplified to allow the application of a feature stereotype model to the spatial foraging trajectory. This simplification will take the form of a warping onto a different type of lattice and a linear interpolation across a plane segment. Clark(1985) developed this type of non-uniform reconstruction as part of a stereo vision problem in which the images sampled were sparse and nonuniform.

A reconstruction is the process of obtaining the exact function that has been sampled, and involves employing information about the underlying function. Reconstruction is similar to interpolation, but interpolation techniques assume nothing about the underlying function, therefore little can be stated about their accuracy. In reconstruction methods the underlying function(s) are assumed to satisfy some constraints from which the reconstruction is determined. Clark's reconstruction algorithm has been applied to this foraging problem and has proved successful in simplifying the feature space on the experimental design that highlights the general areas of search concentration. The way in which his algorithm has been applied here is somewhat different from its original design, consequently some preliminary development of the motivation behind the algorithm is required.

1.2.2 Non-Uniform Sampling in Two Spatial Dimensions

Clark begins development of the warping algorithm by investigating non-uniform sampling in two spatial dimensions.

"The development of a two dimensional non-uniform sampling theory begins with the consideration of the uniform case. The theory behind the reconstruction of functions of two variables from uniformly distributed samples of these functions was developed by Petersen and Middleton (1962). Mersereau (1979) and Mersereau and Speake (1983) have studied the more general problem of processing multidimensional signals that have been sampled on uniform lattices, especially the hexagonal

lattices which are employed in this thesis. " The concern in this case is with signal reconstruction but it is evident that the type of signal processing techniques described by Mersereau and Speake can be extended, applying the results of Clark(1985), to the case of non-uniform sampling. The essentials of the work of Petersen and Middleton (1962) are summarized in theorem 2.2.1. This theorem describes the conditions under which a function of two variables, $f(x)$, can be reconstructed exactly from its samples taken at points on a uniform lattice.

1.2.8 Theorem 1.2.1 The Uniform Two Dimensional Sampling Theorem

"Suppose that a function of two variables $f(x)$ is sampled at points in the infinite sampling set $\{\underline{x}_s\}$ defined by:

$$\underline{x}_s = \{\underline{x} : \underline{x} = l_1 v_1 + l_2 v_2, l_1, l_2 = 0, \pm 1, \pm 2, \pm 3 ..., v_1 \neq k v_2\} \quad 1\,2\,1$$

The vectors v_1 and v_2 form the basis for the sampling lattice defined by the points in $\{\underline{x}_s\}$. Such a sampling lattice is shown in figure 1.2(a) for $v_1 = (2/\sqrt{3}, 0)$ and $v_2 = (1/\sqrt{3}, 1)$. Furthermore let the support of the Fourier transform $F(\omega)$ of $f(x)$ be bounded by the region R in ω space. The spectrum of the sampled function,

$$f_s(x) = \sum_{\{x_s\}} \delta(x - x_s) f(x),$$

is made up of an infinite number of repetitions of the spectrum $F(\omega)$ and is given by

$$F(\omega) = \sum_{\{x_s\}} F(\omega + \omega_s),$$

where the set $\{\omega\}$ is defined by:

$$\omega_s = \{\omega : \omega = l_1 u_1 + l_2 u_2, l_1, l_2 = 0, \pm 1, \pm 2, \pm 3 ..., u_1 \neq k u_2\}$$

and where the frequency domain basis vectors u_1, and u_2, are related to the spatial domain basis vectors v_1 and v_2 by

$$u_1^t v_1 = u_2^t v_2 = 2\pi, \text{ and } u_1^t v_2 = u_2^t v_1 = 0$$

If $F(\omega) = 0$ when $F(\omega + \omega_s) \neq 0$ for every $\omega_s \neq 0$, then the spectral repetitions do not overlap and the following equation holds:

$$f(x) = \sum_{\{x_s\}} f(x_s) g(x - x_s)$$

where $g(x)$ is the inverse Fourier transform of the lowpass filter function $G(\omega)$ defined by:

$$= Q \text{ for } \omega \in R$$

$$G(\omega) = \text{arbitrary for } \omega \notin R, \text{ and } \omega - \omega_s \notin R$$

$$= 0 \text{ for } \omega - \omega_s \in R$$

Q is a constant that is equal to the area of each of the sampling lattice cells and is the inverse of the sampling density. Viewed in terms of the sampling lattice basis vectors, v_1 and v_2, Q is given by

$$Q = \sqrt{|v_1|^2 |v_2|^2 - (v_1^t v_2)^2}$$

For a proof of this result see Peterson and Middleton(1962). "

1.2.4 Mapping Onto a Hexagonal Lattice

"Define $h(x)$ to be a function of two variables, so that $h(x)$ represents $f(x)$ under the coordinate transformation

$$N = M(x) \qquad 1.2.2$$

this means that

$$f(x) = h(N) = h(M(N))$$

Then let the coordinate transformation $M(x)$ be such that the set of non-uniformly spaced samples $\{x_s\}$ is transformed into the uniformly spaced samples $\{N_s\}$ so that the set is the same as that defined in equation 1.2.1. Since $\{N_s\}$ contains points that lie on a regular lattice, the function $h(N)$ can be reconstructed using Theorem 1.2.1. If $h(N)$ is suitably band limited then this reconstruction will be exact. Once $h(N)$ has been reconstructed, $f(x)$ can be obtained by reversing the transform given in equation 1.2.2. This leads to Theorem 1.2.2, which provides the conditions for non-uniform sampling in two spatial dimensions.

1.2.5 Theorem 1.2.2 The Non-Uniform Two Dimensional Sampling Theorem

Define a function of two variables $f(x)$ that has been sampled on the infinite set $\{x_s\}$. If there exists a one-to-one continuous mapping M such that:

$$N = M(x) \text{ and } N_s = M(x_s)$$

and if the function defined by

$$h(N) = f(M^{-1}(N))$$

satisfies the conditions of Theorem 1.2.1 then the following result holds:

$$h(N) = \sum_{\{N_s\}} h(N_s)g(N - N_s)$$

Therefore,

$$f(x) = \sum_{\{x_s\}} f(x_s)g(M(x) - M(x_s)) \quad 1.2.3$$

from Theorem 1.2.1

$$h(N) = \sum_{\{N_s\}} h(N_s)g(N - N_s)$$

Because,

$$M(x_s) = N_s \text{ and } M(x) = N$$

it follows that

$$h(N) = \sum_{\{x_s\}} h(M(x_s))g(M(x) - M(x_s)) \quad 1.2.4$$

The condition that $h(N) = f(x)$, along with the condition that the Jacobian of the transformation M be non zero everywhere, yields

$$f(x) = \sum_{\{x_s\}} f(x_s)g(M(x) - M(x_s)) \quad 1.2.5$$

and concludes the proof of this theorem.

Peterson and Middleton(1962) showed that if $h(N)$ is an isotropic function there is one most efficient type of lattice, and that is a hexagon with characteristic spacing $\frac{\sqrt{2}}{3}$. Here isotropic means that the support of the Fourier transform of the function is centered at the origin and forms a closed disk shaped region.

In the same section it is shown that if this region is denoted as R and defined to be

$$R = \{\lambda : |\lambda| \leq \pi\}$$

then

$$g(N) = \frac{\pi}{3} \frac{J_1(\pi|N|)}{\pi|N|} \qquad 1.2.6$$

where J_1 is a first order Bessel function of the first kind.

An illustration of this lattice can be seen in figure 1.2(a). The values of $h(N)$ are fixed by this lattice geometry, and equation 1.2.6 fixes the values of $M(x)$.

If equation 1.2.5 is to be used as the reconstruction formula, one must, as in the one-dimensional case, know the mapping function $M(x)$ at all sample points and at all other points at which a reconstruction is required. As in the one-dimensional case, once the values of of $M(x)$ are known at the sample points $\{x_s\}$ then it is easy to interpolate $M(x)$ for any x. The problem in the two-dimensional case that is not encountered in the one-dimensional case is that preserving point adjacencies is not a simple problem. The mapping

$$MP : \{x_s\} \text{ onto } \{N\}$$

between sample sets in x and N must be one-to-one continuous everywhere. There are an ∞ of such mappings in two dimensions and no general scheme exists for arbitrarily distributed points so that points that were originally adjacent remain that way. "

Clark (1985) derived an adjacency conserving mapping that employs a well known combination of basis functions which locally satisfy the condition that MP have an inverse MP^{-1}.

1.2.6 Constructing a Generalized Hexagonal Tessellation

There are several word defintions required before commencing the construction of a generalized hexagonal tessellation. They are as follows:

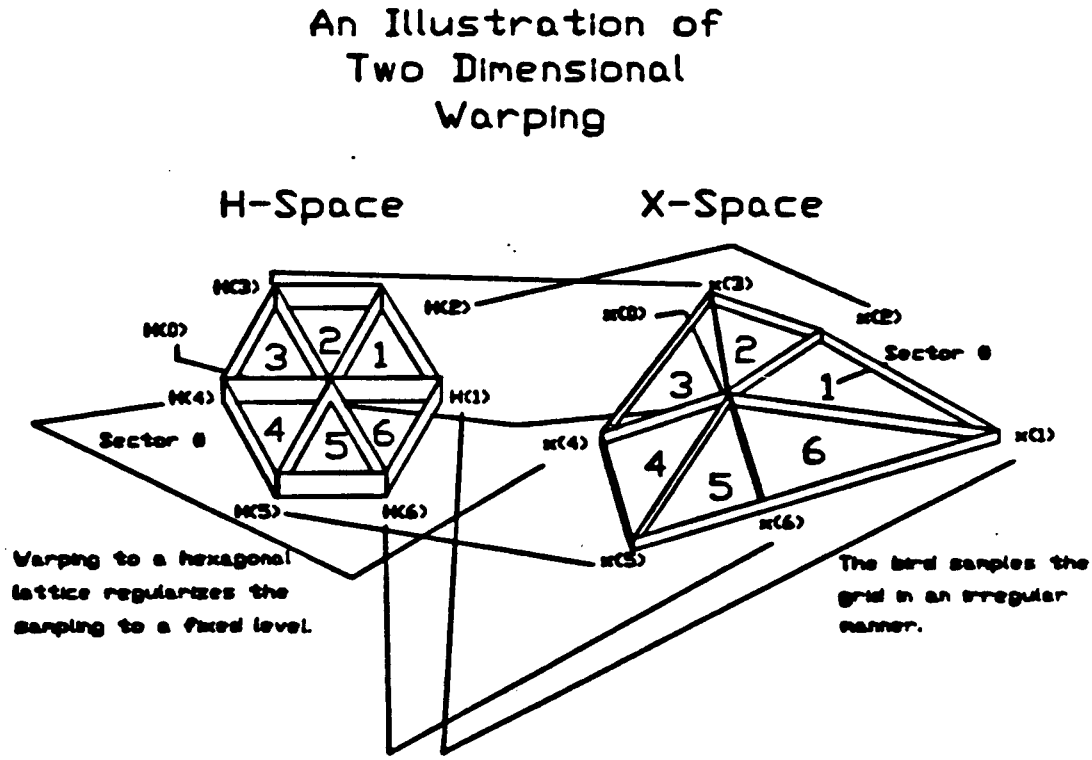

An Illustration of
Two Dimensional
Warping
H-Space
X-Space
H(0)
H(1)
H(2)
H(3)
H(4)
H(5)
H(6)
x(1)
x(2)
x(3)
x(4)
x(5)
x(6)
1
2
3
4
5
6
Sector 0
Sector 0
Warping to a hexagonal lattice regularizes the sampling to a fixed level.
The bird samples the grid in an irregular manner.

figure 1.2(a)

Partition

A partition of a planar region R is a set of line segments, called Links, that divide R into a number of distinct, possibly overlapping subregions. The endpoints of these links are called the Vertices of the partition. There can be no free vertices (i.e., those vertices belonging to only one Link) in a partition except at the boundary of R.

Tessellation

A tessellation is a partition whose regions do not overlap.

Voronoi Tessellation

The Voronoi tessellation of the N plane with respect to the point set $\{N_s\}$ *is the tessellation whose Links consist of points equidistant from any two points* N_i , $N_j \in \{N_s\}$ *and no closer to any other points* $N_k \in \{N_s\}$. *The vertices of the Voronoi tessellation are those points equidistant from three or more points in* $\{N_s\}$ *and no closer to any other point in* $\{N_s\}$. *The subregions created by the Voronoi tessellation contain all points N closer to a given point in* $\{N_s\}$ *than any other point in* $\{N_s\}$.

Dirichlet Tessellation

The Dirichlet tessellation (sometimes referred to as the Delaunay triangulation) is the dual of the Voronoi tessellation. The Dirichlet tessellation can be thought of as the set of line segments connecting the points in $\{N_s\}$ *whose subregions in the Voronoi tessellation of N with respect to* $\{N_s\}$ *share a common Link. An example of Voronoi and Dirichlet tessellations are shown in figure 1.2(b). Further discussion of Dirichlet and Voronoi tessellations can be found in Ahuja and Schacter (1983). It can be seen that the partition created by connecting the points in the hexagonal sampling lattice to their nearest neighbours is a Dirichlet tessellation and as such, the regions created by the partition are non-overlapping triangles. We will denote this particular tessellation by* D_h. "

P-Adjacency

"*Two points are defined to be P-Adjacent if they are vertices of a partition P and share a common Link.*

An illustration of the Spatial warping search strategy

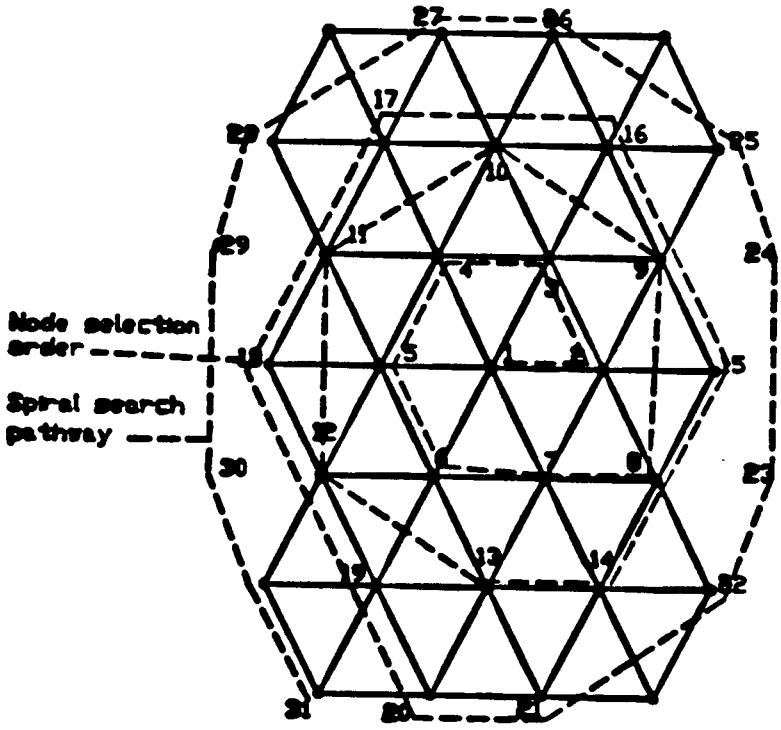


figure 1.2(b)

1.2.7 Adjacency Conserving Partition Mapping (ACPM)

A mapping, like the previously defined MP, is termed an ACPM if it takes a partition having vertex set $\{x_s\}$ into a partition having vertex set $\{N_s\}$ such that the points in $\{x_s\}$ have the same P-adjacency as their images in $\{N_s\}$. As a result of the preservation of adjacency properties, a region in a partition has the same number of Links as the corresponding region in its image under an ACPM. Also, each vertex has the same number of links as its image under an ACPM. Hence, since D_h has triangular regions, the regions of any ACPM, P_x, of D_h are also triangular (although possibly overlapping). The inverse of an ACPM is itself an ACPM.

1.2.8 Generalized Hexagonal Tessellation (GHT)

A tessellation created by applying an ACPM to the tessellation D_h is called a Generalized Hexagonal Tessellation or GHT. All vertices of a GHT are the junction of six Links. As was said at the beginning of this section, once we have a mapping, MP , between the points in $\{x_s\}$ and the points in $\{N_s\}$, we can determine the mapping function, $M(x_s)$ for any x (x is not necessarily a member of $\{x_s\}$) simply by interpolation of the MP values.

The hexagon is made up of six triangular planar regions in P_x. This feature is used to apply a tri-linear plane interpolation function at a given point x. The result of this is to map the region $\{M(x)\}$ to any other arbitrary point in the region. The mapping function is defined as follows:

$$M(X) = \sum_{i=1}^{3} MP(x_i) I(x, x_{(i)_3}, x_{(i+1)_3}, x_{(i+2)_3}) \qquad 1\,2\,7$$

where

$$V(x) = (x_1, x_2, x_3)$$

is the vertex set of the region P_x containing x where (i): denotes (i)modulo(3)+1. The function $I(.,.,.)$ is some interpolation function which results in an invertible N.

The simplest such interpolation between three non-colinear points is a Trilinear Interpolation which fits a planar (vector valued) surface to these three points. This equation describes a plane passing through the points $(x_1^1, x_1^2, 1)$, $(x_2^1, x_2^2, 0)$, $(x_3^1, x_3^2, 0)$. The function has the form

$$I(x, x_1, x_2, x_3) = \frac{(x^1, (x_2^2, -x_3^2) + x^2(x_3^1 - x_2^1) + (x_3^2 x_2^1 - x_2^2 x_3^1))}{\Delta} \qquad 1\,2\,8$$

where

$$\Delta = (x_1^1(x_2^2, -x_3^2) - x_1^2(x_2^1 - x_3^1) + (x_2^1 x_2^3 - x_3^1 x_2^2))$$

and $x = (x^1, x^2)$. "

Clark (1985) then states the theorem which supplies the conditions under which a given one-to-one point set mapping will yield the one-to-one and continuous mapping (or transformation) function that is required for equation 1.2.7 to be valid.

1.2.9 Theorem 1.2.3 Invertibility of a Mapping

"Given the Dirichlet tessellation D_h *on* N *as defined above, and the trilinear interpolation defined by equation 1.2.8 then, if there is an ACPM,* MP^{-1} *such that the image of* D_h, P_x *is a GHT and the points in the set* $V(x)$ *are not colinear, the mapping* $M(x)$ *in equation 1.2.7 is one-to-one and continuous. A complete proof can be seen in Clark (1985) pp.308-309.*

The key condition in Theorem 1.2.3 is that MP^{-1} *be an ACPM, or conversely, that* P_x, *the image of* D_h *be a GHT. Thus, in order to perform the reconstruction of a function for a given sample set* $\{x\}$, *we need to first find the tessellation* P_x *whose image under the mapping* MP *is the hexagonal lattice tessellation,* D_h. *The regions of* P_x *all must have three sides and the vertices of* P_x *must be the junction of six Links.*

It is unlikely that it is possible to create a GHT from all sets of points $\{x_s\}$. *There is no proof of this conjecture, but it can be seen that algorithms for creating GHTs run into trouble trying to order large numbers of points in some cases. When using this construction there is a certain number of points beyond which no points can be mapped without creating an overlapping region or a region with colinear vertices. This, of course, does not constitute a valid proof of his conjecture as there are a number of other ways of trying to construct a GHT from this set of points any one of which may work.* "

As a partial solution Clark presents a method of truncating the reconstruction formula 1.2.5, so that it takes into account only those samples in a finite region about x; it is then not necessary for MP to be one-to-one and continuous everywhere. The mapping function needs to be one-to-one and continuous only over this restricted region. This would mean that we would only need to find a mapping function that creates a partition that is only locally a GHT. In this way it was found that a mapping could be found for any set $\{x_s\}$. The price paid for this weakening of Theorem 1.2.3 is

that the reconstruction is no longer exact, even if $h(N)$ is suitably bandlimited. In practice such a truncation of the reconstruction equation is unavoidable as one can only process a finite number of samples in a finite time.

"Using a finite set of sample points used to reconstruct $f(x)$ to be $\{x_s\}_0$ has some advantages. When only a finite number of terms are used in equation 1.2.7 the resulting value of $f(x_0)$ will not be exact but will be subject to a 'truncation' error term. The truncation error

$$E_t^2(x) = |f(x) - f_R(x)|^2 \leq f_{max}^2 \sum_{\{N_s\}} \frac{1}{6|N - N_s|^2} \qquad 1.2.9$$

The above bound suggests that we make the distances between N and all points not in $\{N_s\}_0$ as large as possible for $N \in \{N_s\}_0$. In other words, the region should consist of the n_s points closest to N. In order for the interpolation of M in equation 1.2.7 to be valid one must stipulate that x be contained in one of the regions of P_{x0}. This also means that N lies in one of the regions of $\{N_s\}_0$. It can be shown that for a given $f(x)$, the localized aliasing error of the reconstruction 1.2.5, decreases as the density of $\{x_s\}_0$ increases. This suggests that one should select the $\{x_s\}_0$ that has the maximum possible density, which insures that, for a given $f(x)$, one will obtain a minimum possible aliasing error, or, alternatively, it will give the maximum allowable bandlimit that a function can possess while still yielding an exact reconstruction. Summarizing this result one can say that the point x at which the function is to be reconstructed must lie within one of the regions of P_{x0}. The density of the set $\{x_s\}$ must be the maximum possible subject to the constraints that the tessellation exists and it can be mapped into the Dirichlet tessellation D_{h0}."

Finding the optimal mapping that jointly minimizes the aliasing and truncation errors is very difficult. Clark developed a heuristic algorithm that successfully implemented these conditions and is near optimal for homogeneous sample distributions found in this foraging experiment.

1.2.10 Implementation of the Two Dimensional Transform

The main motivation behind this algorithm is the assumption that the sample distributions were non-uniform but homogeneous. Using this assumption, the sample points $\{x\}$ can be thought of as arising from the perturbation of the regular sample set, which in this case is the hexagonal lattice. The algorithm assumes that this perturbation is small enough that a given point remains somewhere in a 60-degree sector about its original position. An illustration of this can be seen in figure 1.2(a).

The algorithm begins by trying to find the unperturbed 'original' center point of the hexagonal lattice, denoted as N_0. The perturbed value of this point $x \in \{x_s\}$ that is closest to x is chosen to be the point at which the reconstruction is desired. This point is then to be mapped to N_0 using $n_s = 7$ support points. Once this first point is chosen the other $n_s - 1$ points are found in a similar manner as the search moves around each of the 60-degree sectors. The search then spirals outward until all points have been mapped or a boundary reached. An illustration of the spiral search can be seen in figure 1.2(b).

A sample of a raw grid can be seen in figure 1.2(c) and the resulting regularized grid can be seen in figure 1.2(d).

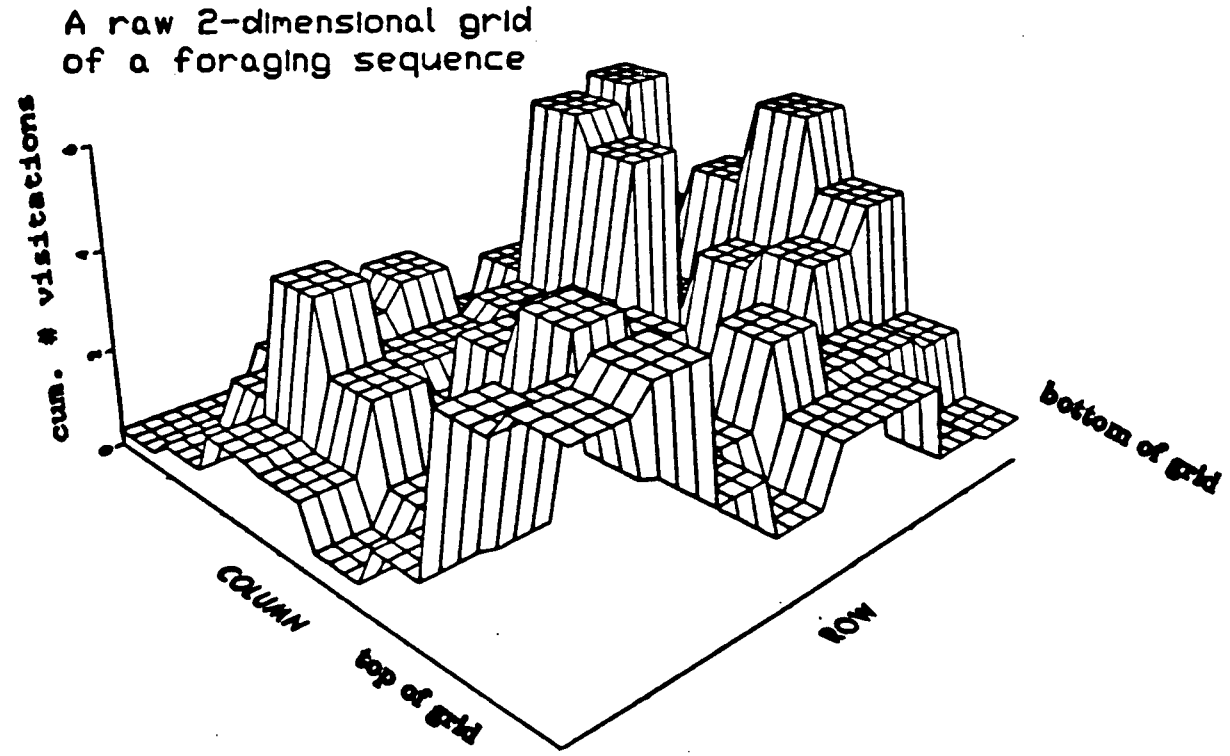


figure 1.2(c)

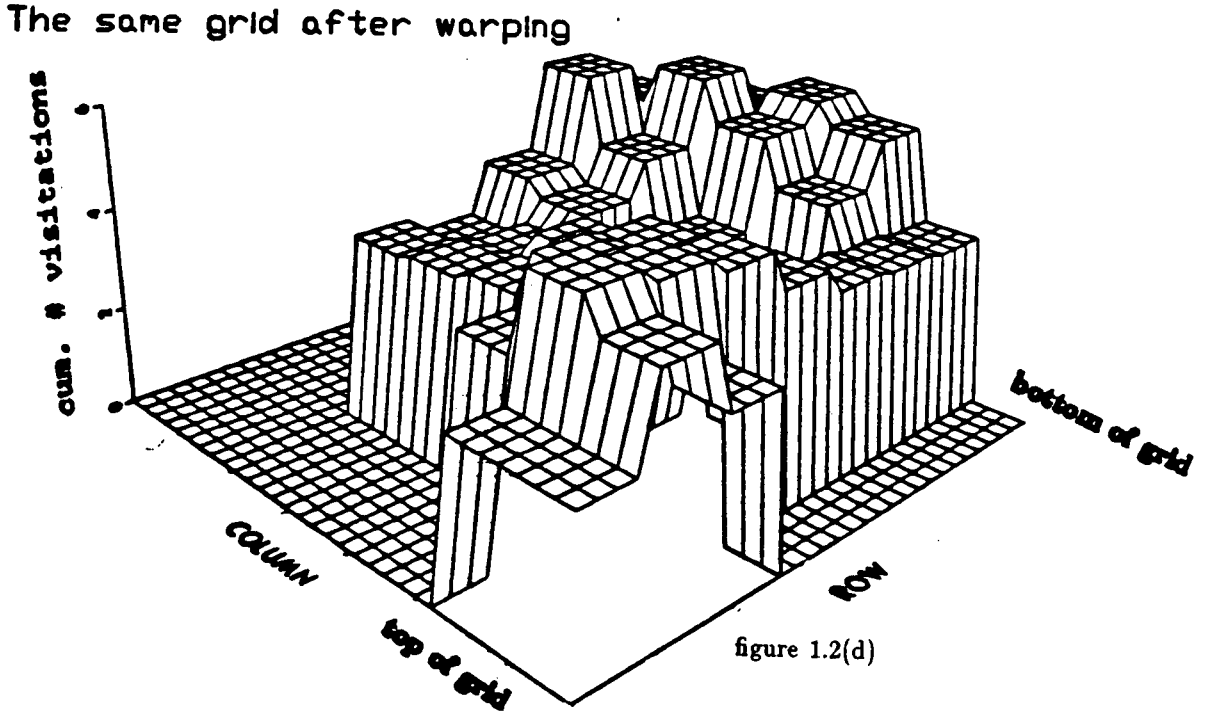


figure 1.2(d)

figure 1.2(c,d)

1.2.11 Summary of Appendix 1.2

- The concept of two-dimensional warping was introduced.

- The hexagonal lattice was introduced as a basis for the warping.

- An adjacency conserving partition mapping was introduced

- These concepts were implemented, applying them to actual foraging sequences.

References

Ahuja, N., Schacter B., 1983. Pattern Models, John Wiley and Son, New York.

Bellman, R., 1957. Dynamic Programming. Princeton N. J., Princeton University Press.

Bobrow D., Winograd T., 1977. An overview of KRL, a knowledge representation language, Cognitive Science, vol 1, pp 3-46.

Brown, M. K., Rabiner, L. R., 1982. An adaptive, ordered, graph search technique for dynamic time warping for isolated word recognition. IEEE Trans. Acoustics, Speech and Signal Processing, vol ASSP-30, No. 4.

Chandrasekaran, B., 1983. Expert Systems: Matching Techniques to tasks, in Artifical Intelligence Applications in Business. W. Reitman editor, Ablex Publishing Corporation, Norwood, N.J.

Chomsky, N., 1963. Formal properties of grammers. In R. Luce, R. Bush and E. Galanter, eds, Handbook of mathematical psychology, vol 2, New York, Wiley.

Clark, J., 1985. Multi-resolution stereo vision with application to the automated measurement of logs. Ph.D. thesis in the faculty of Electrical Engineering. The University of British Columbia.

Fikes, R., Kehler, T., 1985. The role of frame-based representation in reasoning. Communications of the ACM, vol 28 number 9, pp 904-920.

Gass L., Montgomerie R., 1981. Hummingbird Foraging Behaviour: Decision Making and Energy Regulation. *In* Foraging Behaviour: ecological, ethological and psychological. Chapter 8 in Kamil and Sargent eds., Garland Publishing Company.

Gass, L., Sutherland, D., 1985. Specialization by territorial hummingbirds on experimentally enriched patches of flowers: energetic profitability and learning. Can. J. Zool. 63: 2125-2133.

Gould, J., 1986. Spatial pattern learning by honey bees, Animal Behaviour, vol 34, pp 990-997.

Gould, J., Marler, P., 1987. Learning by instinct, Scientific American, January, vol 12, no 256, pp 74-85.

Havens, W., 1978. A procedural recognition model for machine intelligence. Ph.D. thesis, in the Department of Computer Science. The University of British Columbia.

Kamil, A., Yoerg., S., 1982. Learning and foraging behaviour, in Perspectives in Ethology, Bateson, P., and Klopfer P. eds, vol 5, Plenum press.

Kleene, S. C., 1952, Introduction to Metamathematics, Van Nostrand, Princeton, N.J..

Kowalik, J.,S., 1986. Coupling Symbolic and Numeric Computing in Expert Systems, North Holland, New York.

Kowalski, R.A., 1982. Logic as a computer language, in Clark, K. L., & Tarnlund, S.A., eds. Logic Programming. Academic press.

Krebs, J., Stephens, D., and Sutherland, W., 1983. Perspectives in optimal foraging, in Perspectives in ornithology. Cambridge University Press. pp 165-216.

Lubell, D., 1966. A short proof of Sperner's lemma, J. Combinatorial Theory vol 1, p 299.

McNamara, J., Houston, A., 1986. The common currency for behavioural decisions, The american naturalist, vol 127, no 3, pp 358-377.

Mangel, M., Clark, C., 1986. Towards a unified foraging theory, Ecology, vol 6, pp 1127 - 1138

Mersereau, R., 1979. The processing of hexagonally sampled two-dimensional signals. Proceedings of the IEEE, vol 7, pp 930-949.

Mersereau, R., Speake, T., 1983. The processing of periodically sampled multidimensional signals. IEEE Transactions on Acoustics, Speech and Signal Processing, vol 31, no 1, pp 188-194.

Minsky M., 1975. A framework for representing knowledge, in P. Winston ed., The Psychology of computer vision, McGraw Hill, New York, pp. 211-277.

Myers, C. S., Rabiner, L. R., 1981. A comparative study of several synamic time warping algorithms for connected-word recognition. Bell System Technical Journal, vol 60, No 7.

Peterson D., Middleton D., 1962. Sampling and reconstruction of wave-number limited functions in N-dimensional Euclidean spaces. Information and Control, vol. 5, pp 279-323.

Pyke, G., 1984. Optimal foraging theory: a critical review, Ann. Rev. Ecol. Syst., vol 15, pp 523-575.

Rabiner, L. R., Rosenberg, A. E., Levison S. E., 1978. Considerations in dynamic time warping algorithmns for discrete word recognition. IEEE Trans. Acoustics, Speech, and Signal Processing, vol ASSP-26, No 6.

Roberts, B. K., 1984. Digital processing of somasensory evoked potentials applied to early diagnosis of multiple sclerosis. MASc thesis in the faculty of Electrical Engineering. The University of British Columbia.

Shank R., Abelson R., 1977. Scripts, plans, goals and understanding. Hillside, New Jersey, Lawerence Erlbaum.

Shapiro, E., 1983. Algorithmic program debugging. MIT Press.

Shettleworth, S., 1983. Scientific American, vol 247, pp 102-110.

Sperner, E., 1928. Ein Satz über Untermengen einer endlichen Menge, Math. Zeitscr. vol 27, pp 544-548.

Sutherland, G., 1986. The role of spatial pattern learning in hummingbird foraging, M.Sc. Thesis in the department of Zoology, University of British Columbia.

Walker, A., 1986. Knowledge systems: Principles and practice, IBM J. Res. Develop, vol 30 no 1, pp 1-12.

Watanabe, S., 1969. Knowing and Guessing, Chapter 7, John Wiley and Sons, N.Y..

Watanabe, S., 1985. Pattern recognition: human and mechanical, John Wiley, New York, pp 509-153.

Winograd, T., 1982. Language as a cognitive process. Addison Wesley, Reading Mass. .