EXPLORATORY ROBOT LEARNING OF
SENSORY-MOTOR INTEGRATION

W. SCOTT SPANGLER

MAY 1988       A188-79
EXPLORATORY ROBOT LEARNING
OF SENSORY-MOTOR INTEGRATION

by
William Scott Spangler, B.S.

THESIS

Presented to the Faculty of the Graduate School of
The University of Texas at Austin
in Partial Fulfillment
of the Requirements
for the Degree of
Master of Arts

THE UNIVERSITY OF TEXAS AT AUSTIN

May 1988
ACKNOWLEDGEMENTS

My interest in Critter Learning originated from a term project assigned in Dr. Bruce Porter's Machine Learning class in the spring of 1987. I am indebted to him for stimulating my interest in this subject. The work which I and the other members of my research group, Yung-Tai Byun and David Pierce, did that spring was indispensable in revealing the difficulties of the Action Learning Critter problem, as well as pointing the way towards the eventual solution outlined in this thesis. I am grateful to Yung-Tai also for the use of his critter simulation program and for his help in tailoring it to the needs of this research.

I would like to thank my family for their moral support and the General Motors Knowledge Engineering Group for the financial support of my graduate education.

Thanks go to Dr. Robert Causey for taking the time to be a second reader of this thesis. And finally special thanks to my advisor Dr. Benjamin Kuipers for his insightful comments and expert guidance during the entire course of this work.

W. Scott Spangler

The University of Texas at Austin
May, 1988
ABSTRACT

EXPLORATORY ROBOT LEARNING OF
SENSORY-MOTOR INTEGRATION

by

William Scott Spangler, B.S.
Supervising Professor: Benjamin J. Kuipers

This thesis describes an algorithm for mobile robot learning of sensory-motor coordination. The goal is to learn how to characterize the effect that taking a particular movement action will have on a robot's sense input, given the initial sensory situation. The concepts are learned from the sensory results of taking random actions from within a training environment (Learning from Examples), but these concepts are general enough to apply to unseen environments. The learning algorithm is implemented and tested for a particular type of simulated sonar sensing robot.
Contents

1 Introduction
   The Map Learning Critter .................................. 1
   The Action Learning Critter ............................... 2
   The Importance of this Problem ............................ 4
   Comparison to Other Research .............................. 5

2 Critter Specifications
   The Sensory-Motor System .................................. 6
   The Sonar Critter .......................................... 7

3 Simplifying Assumptions ................................ 10

4 General Approach ........................................ 12

5 Rotation Learning
   Motivation .................................................. 14
   Rotation Learning Algorithm ............................... 16
   Results ...................................................... 18
   Analysis .................................................... 21

6 Translation Learning
   A Definition for $\Delta S$ ................................ 23
   An Example ................................................ 25
   The Tree Generation Algorithm ............................ 26
Results .......................................................... 29
Analysis of the Algorithm ................................. 32

7 Imaginary Actions ............................. 38

8 Testing ................................................. 43
Quantitative Tests ...................................... 44
Qualitative Tests ....................................... 45

9 Conclusions ..................................... 52

10 Much Ado about Nothing ............... 54
Some Questions about Knowledge .................... 54
Motivational Systems .................................. 56
Knowledge about Actions ............................ 57
Interrelation of Senses ................................ 58
Generality vs. Efficiency ............................ 59

11 Areas for Future Research ............... 61
Eliminating A Priori Assumptions ................... 61
Qualitative Understanding of Actions ............ 63
Generalization over Actions ......................... 64

12 Glossary of Notation ....................... 65
Chapter 1

Introduction

This research developed as an outgrowth of the Critter Problem, proposed by Ron Rivest in 1984. A robot or animal (referred to as the critter) is born tabula rasa into an environment. The critter has a set of discrete actions (i.e. movements) $A_1 \ldots A_n$ which it can perform and a set of sense devices $D_1 \ldots D_m$ which are its source of information about the environment. The critter has two goals: (1) to learn the concepts which describe the effect of each of its $n$ actions; (2) to explore and characterize the environment. We refer to the first of these goals as Action Learning, and the second as Map Learning. [Kuipers 85].

The Map Learning Critter

The second goal, that of exploring and characterizing the environment, is currently being studied by Y.T. Byun and Benjamin Kuipers. Their approach is to create a topological model of the environment consisting of nodes and arcs: the nodes representing distinctive sense impressions and the arcs representing the set of actions taken between such sense impressions [Kuipers 87]. This qualitative approach to map learning differs fundamentally from the Cartesian coordinate representation schemes adopted in other robot exploration research. It has the advantage of being more nearly analogous to
the way humans understand their environment, as well as helping to control cumulative error in sense input and motor calibration.

In order to explore an unknown environment, the map learning critter needs a local control strategy—some way of selecting actions which will allow it to move about safely and intelligently, and also facilitate discovery of distinctive places. Learning the sensory-motor coordination which is necessary for an effective local control strategy is the domain of the Action Learning critter and the goal of this research.

The Action Learning Critter

The primary task of the Action Learning Critter is to learn concepts which describe the sensory change caused by an action, without using or being given any sort of map or description of the environment. To be effective the action concepts learned in the training environment must be applicable to different, unexplored environments.

We believe that the following scenario represents successful Action Learning given a critter with some sensory-motor system, a training environment, and a testing environment:

1. The critter takes actions in a training environment, observing the changes in sense data which result and learning from these observations.

2. The critter is placed in a new unexplored environment and given some well-defined sensory goal, such as move away from the nearest boundary, or maintain a particular sensor at a specific sensory value.

3. The critter successfully accomplishes the goal in the new environment using the knowledge gained in the training environment.

Figure 1 illustrates the route taken by a distance sensing critter as it accomplishes first the goal of moving away from the nearest boundary and second
the goal of staying a constant distance from the right hand wall.

![Diagram](image)

Figure 1: Successful accomplishment of two well-defined sensory goals

The difficulty involved in Action Learning is that all the information the critter has to work with is what it gathers from its sense devices. The critter knows nothing initially about the arrangement and meaning of its sense devices, the shape of its environment, or its own position within that environment. And once an action has been taken from some initial position, the critter has no way to return to the original position to try a different action and directly compare the results.

Stated more formally the Action Learning Critter problem is the following:

- **Given:** A set of action instances taken by a single critter from random positions in a training environment, where the sense data from each instance is stored in the following format:

  \[(A, S_0) \rightarrow S_f\]

  $A$ being an action, $S_0$ an initial sense impression, and $S_f$ the final sense impression after the action has taken place.

- **Learn:** Concepts which approximately predict the sensory result ($S_f$) of taking any given action from some initial sensory situation (which
may be entirely different from any previously encountered $S_0$). The approximation should improve with increases in the size of the training set, and be accurate enough to facilitate choosing actions to achieve simple sensory goals.

- **Scope of Concepts:** Concepts refer only to the critter sensory-motor system from which they were learned. Concepts should, however, apply to unseen environments and situations.

- **Limitations:** The sensory input data needs to be virtually error free. The sense devices are assumed to be directional in nature. These and other limitations are discussed in the next chapter.

### The Importance of this Problem

The Action Learning Critter problem is important for other reasons aside from its role in facilitating the Map Learning Critter. All robots need some form of sensory-motor coordination if they are to act intelligently in an unknown environment. As the technology improves and the complexity of the sense devices and the range of possible actions increases, it becomes more and more difficult to impart this information with static programming methods. Eventually it will become impractical to program into a robot all necessary sensory-motor knowledge, before it ever takes its first action. Some sort of sensory-motor learning will be needed in complex robots, just as it appears to be necessary in higher order animals.

The Action Learning Critter also has implications in the field of Cognitive Psychology. Though it has been fairly well established that human infants do some degree of sensory-motor learning, it is not yet well-understood exactly how much is learned as opposed to being instinctive. It is also not at all clear how the mind represents such information. Solving the Action Learning Critter problem might provide some insight into these mysteries.
Comparison to Other Research

The original Map-Learning Critter described in [Kuipers 85], learned to interpret its actions by performing experiments and examining the periodicity of the sense impressions, the result being the classification of actions as "turn-like," "travel-like," and "others." The critter described in this research learns to predict the approximate sensory change which results from taking each of its actions. This not only helps it in classifying actions as "rotations" or "translations," but also gives it the ability to avoid boundaries and seek out desirable sensory situations.

Ron Rivest and Robert Schapire have done work on learning in a deterministic finite state environment—a problem which is also being analyzed and formalized by Uriel Feige and Adi Shamir as "Learning in Permutation Groups." Their approach is for the learner to construct a model of the environment which correctly predicts the result of any proposed sequence of actions [Rivest 87]. This approach is limited to sensory-motor systems for which the total number of possible actions and sense inputs is relatively small.

Tom Mitchell is applying an explanation based learning methodology to the problem of robot arm manipulation [Mitchell 88]. In this approach, however, a great deal of sense interpretation is done for the robot before learning even begins. The object in Critter Learning is to begin with almost no knowledge or interpretation of sense input. Our approach has the advantage of generality with respect to differing sensory-motor devices and arrangements, and can also more readily deal with systematic errors in motor calibration or sense device orientation.

As far as Machine Learning research is concerned, we will build upon many of the ideas and algorithms of Quinlan (Learning by Examples) and Michalski (Conceptual Clustering). We will not present any radically new procedures in these areas, but we will show how the existing techniques can be adapted to apply to this particular learning domain.
Chapter 2

Critter Specifications

This chapter describes the limits of the applicability of this research. Our general concept of a critter is a very simple sort of mobile robot. Nothing changes in the environment except when the critter takes an action and the only actions the critter may take are movements, relative to the environment. The critter gathers information about the effect of these actions using its sense devices. Section 2.1 describes the limitations which are placed on these actions and sense devices, and section 2.2 gives an example of a particular type of sonar critter which meets these specifications.

The Sensory-Motor System

This research applies to a particular type of sensory-motor system which has the following properties:

- Directional Sensing Devices – The sense input devices, whatever their form may be, are directional in nature—meaning that each sense device always points in a particular, constant direction relative to the critter. Each sense device returns a single integer value before and after each action.

- Discrete Movement Actions – The only actions which the critter has available to it, are those which cause displacement. The critter has
only a finite (and relatively small) set of basic movement actions. Each action has a limited duration, and the displacement caused by taking a single action is small relative to the size of the environment.

An example of such a sensory-motor system, the sonar critter, will be described in the next section, but a sonar sensing critter is certainly not the only type of robot which fits the above specifications. Consider a mobile robot with an electro-optic vision system where each pixel of the optic sensor returns an integer corresponding to a gray-scale value. We could apply our Action Learning algorithm to this sensory system by considering each pixel to be a distinct sensory device. The critter could then learn in what manner taking a certain rotation action causes the visual image to shift, or how going forward causes image magnification (if the camera is facing forward).

Information from tactile sense devices could also be used by an algorithm similar to the one to be described. In fact the general form of the learning algorithm should apply to any exploratory, mobile robot which has multiple sensing devices oriented to observe the environment in differing directions, or even a single sense device which rotates to observe several different directions between each action. Such directional sensing is, for all practical purposes, a prerequisite for any kind of intelligent exploration of a spatial environment.

The Sonar Critter

This research will concentrate on a particular type of critter called LASI (Learning Actions from Sonar Input) which lives in a two dimensional world consisting of areas enclosed by boundaries. The critter can move around in this world using two tractor-type chains (like a tank), and it can sense boundaries using sixteen “sonar range finders” which are mounted on the critter and oriented at 22.5 degree intervals in a 360 degree radius. These
are numbered from 0 to 15 clockwise starting with the sonar which faces directly forwards.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{critter_sonar_arrangement.png}
\caption{The Critter’s Sonar Arrangement}
\end{figure}

LASI’s movements are characterized by energy pairs \((x, y)\) where \(x\) is the amount of power supplied to the left chain and \(y\) is the amount of power supplied to the right chain, \(x\) and \(y\) being integers ranging from 5 to -5. Supplying an energy pair will cause LASI to change position and/or orientation in a manner corresponding to the magnitudes of \(x\) and \(y\). For instance, the input pair \((2, 2)\) will move the critter directly forward some distance, while \((4, 4)\) will do the same only twice as far. The input \((3, 2)\) will move the critter forward and to the right, shifting its orientation clockwise. The action \((-3, 3)\) will rotate LASI counterclockwise without changing LASI’s position at all. Each of LASI’s sonar range finders may take on any value between 1 and 220, indicating the distance from the critter to the nearest boundary in that sonar’s direction. If there is no boundary within range of a sonar, then the sonar returns the value 220. The sonars return values only before and after each critter action.

The choice of this particular critter was motivated by several factors. First, the range of possible sense inputs and motor outputs is large enough to be interesting without being intractable. Second, neither the environment nor the critter are computationally expensive to simulate. And third, the sonar critter closely resembles the sort of mechanical robot which can be
built cheaply with hardware that is currently available. On this last point, however, it should be noted that the critter simulation will ignore many real-world problems, such as random errors in sonar range finder inputs, systematic errors caused by poor boundary reflection of sonar signals, or hardware inconsistencies such as tread malfunction or slippage.

It should be noted that even though this particular critter simulation is technically a discrete sensory-motor system (since sense inputs and motor outputs are integers), the range of possible sense inputs is much larger than a finite state scheme like that proposed by Rivest et. al. could handle. If we assume that each distance sensor may take on any value within 20 pixels of its nearest neighbors (a restriction which limits us to reasonably continuous environments) then there are a total of $40^{16}$ or approximately $10^{26}$ possible sense inputs. Limiting LASI to one particular environment might reduce this number dramatically, but that would defeat the whole purpose of an exploratory critter.
Chapter 3

Simplifying Assumptions

Ideally we would like to design a critter which could learn the effect of actions given no initial information about the nature of its sense devices. But in order to focus attention on the more interesting aspects of Action Learning, we assume that some information is given to the critter \textit{a priori}, before learning even begins. The following is a description of this information along with some other simplifying assumptions.

1. The critter knows that the 16 sense devices all have the same form and function. Each returns a single integer value before and after every action.

2. The range of possible values for each sense device is given to the critter \textit{a priori}.

3. The critter knows that sense device $i + 1 \mod 16$ is oriented just to the right of sense device $i$. It is assumed that each sense device points in a particular direction (relative to the critter) which is constant and distinct from any other sense device. The exact relative directional difference between sense devices is not known to the critter.

4. The environment and the critter’s position within it does not change between critter actions.
5. It is assumed that taking any particular action, $A_k$, should always change the critter’s position in the same manner (though the change in sense input which results from the change in position will probably not be the same). The environment does not itself change as a result of a critter action.

6. Sense input is virtually error free.

The first three assumptions above may appear incompatible with the spirit of *tabula rasa* action learning. This issue will be discussed at greater length in Chapter 10. For now we justify these assumptions on the grounds that in most robots this knowledge is usually so simple to program in (e.g. in the case of assumption 3, by numbering the sense devices in some intelligent way and then writing a function which returns the number(s) of the neighbor(s) of a particular sense component given that sense component’s number) that it would seem impractical and uninteresting to have the critter learn such concepts. Nevertheless, possible algorithms for acquiring such knowledge are discussed in Chapter 11.

The last three assumptions are idealizations from the real world. Eliminating any of these assumptions comprises an interesting and important research project in its own right—but one which lies outside the scope of this work.
Chapter 4

General Approach

This research applies Machine Learning techniques to the Action Learning Critter problem. In particular the Action Learning Critter is treated as a special case of Learning from Examples, a term which is used to describe algorithms which derive general concepts from a training set of specific concept instances [Carbonell 83]. The challenge is to achieve a reasonable degree of concept generality while still retaining the important features of each training instance. In critter learning this poses an especially difficult problem. Each training instance is given in the form \((A, S_0) \rightarrow S_f\), where \(S_0\) is the vector of values received from sense devices \(D_1 \ldots D_m\) before the action \(A\), and \(S_f\) is the corresponding vector after \(A\) has been executed. Since there is nothing in the specifications of the critter problem to guarantee that any two actions will be sufficiently similar to warrant including more than one in a single concept, we have to focus on generalizing over sense vectors. But without having any knowledge of the environment how is one to say whether any two given sense vectors are similar or dissimilar?

The approach which our critter learning algorithm uses to solve the generality problem is the following:

- The sense input change caused by taking an action is broken down into a rotation component and a translation component. This will be explained further in the next section.
• Each action instance is written in the form \((A, S_0) \rightarrow \Delta S\) instead of \((A, S_0) \rightarrow S_f\), where \(\Delta S\) stands for a characterization of the change in sense input from \(S_0\) to \(S_f\). We will explain the form of this characterization in more detail later.

• Action learning is not done at the sense vector level, but at the sense vector component level. Learning at a lower level increases the generality of the concepts, since concepts about sense vector components can apply to a wide range of sense vectors—even those obtained from environments very different from that in which the original concept was learned.

These three techniques effectively reorganize the data provided by the training instances in such a way as to maximize the amount of previous experience that can be used in comprehending a new situation.

At this point it might be wise to dispel any remaining illusions concerning what sort learning is actually involved here. The learning algorithm presented in this thesis is really little more than an information organizer. It takes raw sense data as input and then rewrites it in a form that will make it applicable to new situations. Certainly, this type of learning is not particularly glamorous, but it is learning nonetheless. In fact it seems likely that most learning done by animals (especially sensory-motor learning) consists mainly of information organization of some sort or other, as raw sense data is distilled into a form which can be remembered and applied to new situations.
Chapter 5

Rotation Learning

When a critter executes one of its actions, a change will take place in the critter's position, in its orientation, or in both position and orientation. Rotation Learning attempts to dissociate (as much as possible) the sense change caused by a change in critter orientation from that caused by a change in position, and to describe this change in terms of the directional spacing between sense input devices.

Motivation

In general, the sense input change which results from a change in critter position depends on the shape of the local environment relative to the critter. But the sense change which results from a change in orientation (whether or not it is accompanied by a change in position) may be, in whole or in part, expressed in terms which are independent of environmental conditions. For instance, assume the "turn-around" action rotates LASI 180 degrees without changing the critter's position. A simple concept which describes $S_f$ given $S_0$ for this action is the following:

$$S_f[i] = S_0[i + 8 \mod 16]$$

Figure 3 illustrates this.
Figure 3: The result of executing a turn-around action

I refer to concepts of this form as rotation concepts. Generally rotation concepts indicate a correlation between the sense input of device $D_i$ after a particular action has been taken, to some sense device $D_j$ before the action was executed. The goal in Rotation Learning is to use such concepts to describe the orientation change of the critter caused by each action in terms of the orientation relationships between the critter's sense devices. Rotation Learning may be done for all of LASI's actions, not just those of the form $(-x, x)$.

For a particular action, $A_k$, Rotation Learning seeks to discover for each component of the final sense impression $S_f[i]$, the component of the initial sense impression $S_0[j]$ whose value in relation to its nearest neighbors most often decides the value of $S_f[i]$ in relation to its neighbors. This is determined from a set of random examples of the action $A_k$ taken by the critter from within some training environment, using the algorithm described in the next section.

The goal of Rotation Learning as described above may seem a bit mysterious. Intuitively we are trying to find the components of $S_0$ which have
the most influence in determining the value of $S_f[i]$ for some value of $i$ and some action $A_k$. This allows us to focus our attention in Translation Learning on those components—greatly reducing the amount of training instances needed to learn effectively.

**Rotation Learning Algorithm**

Rotation Learning is accomplished separately for each action and sense device. The critter collects a set of training instances each of the form $(S_0, S_f)$ for each action starting from random positions in the environment. For a particular sense device $D_i$ we let

$$l = i - 1 \mod 16$$
$$r = i + 1 \mod 16$$

so that $D_l$ and $D_r$ are oriented just to the left and right of $D_i$. The function $simplify(S, i)$ is defined in the following manner:

- If $S[i] = 220$ then $simplify = \infty$
- If $S[l] < S[i]$ and $S[i] > S[r]$ then $simplify = \cap$
- If $S[l] > S[i]$ and $S[i] < S[r]$ then $simplify = \cup$
- If $S[l] < S[i]$ and $S[i] < S[r]$ then $simplify = \nearrow$
- If $S[l] > S[i]$ and $S[i] > S[r]$ then $simplify = \swarrow$
- If none of the above conditions apply then $simplify = 0$

The $simplify$ function classifies the value of a particular sense component into one of six categories. These categories describe the relationship between sense device $D_i$ and its two nearest neighbors. The symbols $\cap$, $\cup$, $\nearrow$, and $\swarrow$ were chosen to give a pictorial representation of this relationship (i.e. the $\cup$ symbol indicates that $D_i$ is the minimum of the three values).
The function simplify gives us a representation of the local contour of the sense impression immediately surrounding $D_i$. We expect the sense contour to remain fairly constant for small changes in critter position, therefore when we detect a consistent shift in sense contour from one sense input device to another across different instances of the same action, we can attribute this shift to the change in critter orientation caused by the action.

Now assume that we want to learn the rotation for sense device $D_i$ when taking some action $A_k$. We simplify the $S_0$ vectors in the training instances using the simplify function on each element of each vector. Each $S_0$ can now be thought of as a vector of attribute values, where each position in the vector is an attribute. The task now reduces to that of finding the best attribute to classify the value of simplify($S_f[i], i$) over the set of training instances. This is accomplished using Quinlan's information theoretical approach for choosing attributes [Quinlan1 86]. The component of $S_0$ which is chosen (referred to by the index $j$) should minimize $E(j)$, where:

- $E(j) = \sum W(v) \times M(v)$, where $v$ takes on the six values of the simplify function.

- $W(v)$ is the number of training instances where simplify($S_0, j$) = $v$

- $M(v)$ is $\sum -p \log p$ over the set of six simplify($S_f, i$) values where $p$ is the proportion of times the value occurs in the set of training instances where simplify($S_0, j$) = $v$

Quinlan's algorithm gives us the component of $S_0$ which most accurately foretells the value of $S_f[i]$ over a set of training instances.

It is usually impossible for the directional difference between sense components related through Rotation Learning to exactly match the orientation change caused by the action. This is due to the fact that actions seldom, if ever, rotate LASI by some multiple of 22.5 degrees (which is the directional spacing of the sense devices). Rotation Learning merely reveals which component of $S_0$ is of greatest importance in predicting $S_f[i]$.
Results

The result of Rotation Learning is a vector of 16 \((i,j)\) pairs for each of the 27 different actions, where \(i\) refers to a component of the \(S_f\) vector, and \(j\) names the component of \(S_0\) which most nearly corresponds to it for that action. Two examples of such vectors, are shown below:

Action \((3 3)\): \(((0 0) (1 1) (2 2) (3 3) (4 4) (5 5) (6 6) (7 7) (8 8) (9 9) (10 10) (11 11) (12 12) (13 13) (14 14) (15 15))\)

Action \((-3 3)\): \(((0 13) (1 14) (2 15) (3 0) (4 1) (5 2) (6 3) (7 4) (8 5) (9 6) (10 7) (11 8) (12 9) (13 10) (14 11) (15 12))\)

For the straight-ahead action \((3 3)\), \(j = i\) for every sense device, indicating no rotation component for that action. On the other hand, \(j = (i - 3) \mod 16\) for the rotation action \((-3 3)\), indicating a rotation of approximately 3 sense device spacings (3 \(\times\) 22.5 degrees). If we graph the two vectors on an \(i\) vs. \(j\) plot we get the following:

![Graphs](image)

Figure 4: Rotation graphs for the actions \((3 3)\) and \((-3 3)\)

The graph on the right is identical to that on the left, except that each point is shifted 3 units downward (along the \(j\) axis). This produces a wrap-around effect because of the modulo 16 arithmetic.
On the next two pages rotation graphs are shown for the rest of the critter's 27 actions. The training set for each action consisted of 60 action instances taken from different positions inside a rectangle. The change in critter orientation caused by each action is reflected in the amount of shift of the plotted points of away from the \( j = i \) line. Thus the greatest shift can be seen for the actions, (5 -5) and (-5 5), while little or no shift occurs for the straight-ahead actions: (5 5), (4 4), (3 3), (2 2), and (1 1).

Ideally we would like to see the plotted points form a straight line for every action (since the sense devices are spaced symmetrically), but unfortunately this is not always the case. There are two reasons for the discrepancies exhibited in some of the rotation graphs, which are not the fault of the algorithm.

- **Non-integral Rotation** – The critter actions do not necessarily rotate it an integral number of sense devices (i.e. some multiple of 22.5 degrees). This is undoubtedly the problem for the action, (5 -5), which causes the critter to rotate approximately 170 degrees. Thus the graph for this action shows some points shifted \( 8 \ j \) units (180 degrees), while others are shifted only \( 7 \ (157.5 \ degrees) \).

- **Translation Shifts** – As the critter moves forward there is a tendency for the image on the right side to shift slightly clockwise while the image on the left side shifts counterclockwise. This is especially apparent in the graph of the action (4 5) where the points are plotted along two distinct lines: \( j = i \) and \( j = i - 1 \), for the sense devices on the left and right side respectively. Since we stipulated in Chapter 2 that critter actions cause displacements which are small relative to the size of the environment (as is the case for LASI's actions), we do not expect translation shifts to be a serious impediment to diagnosing the orientation change caused by an action independent of the position change. And in fact, in the results which are shown the translation shift never causes a discrepancy of more than one \( j \) unit.
Rotation Learning Results
Figure 5: Rotation Learning Graphs

Analysis

All of the results shown above reflect an accurate representation of the actual rotation caused by each action, to within 1 sense component (22.5 degrees), as can be seen from the fact that none of the plotted points on the graphs are more than a single $j$ unit off-line from the other points. We have thus accomplished the original goal of Rotation Learning.

One might doubt the necessity of using Quinlan’s algorithm for choosing attributes here. It should after all be just as accurate to choose the component of $S_0$ whose simplified value is most often the same as the simplified value of $S_f[i]$. But this would be making the a priori assumption
that each of the sense devices was equally spaced (directionally) around the sensory ring. Consider a critter whose sense devices are oriented as shown in the figure below:

![Diagram](image)

Figure 6: An asymmetric sonar critter before and after a turn-around action

Before the turn-around action \( \text{simplify}(S_0, 0) = \nearrow \) as pictured on the left. After the action \( \text{simplify}(S_f, 8) = \cap \) as shown on the right. If correlation were measured simply by equality, such instances of the turn-around action would be evidence against correlating sense device 0 with sense device 8. But Quinlan's algorithm would perceive that whenever \( \text{simplify}(S_f, 8) = \cap \), then \( \text{simplify}(S_0, 0) \) would almost always be one of three values: \( \cap, \nearrow \), or \( \searrow \), which would tend to indicate a connection between these two sense components. Because Quinlan's algorithm detects more than the simple equivalence correlation, it can be effective even when the sense devices are not spaced symmetrically.
Chapter 6

Translation Learning

The goal in Translation Learning is to discover the sense input change which occurs due to a change in position, making allowances for the known change in orientation (as given by Rotation Learning). The general approach to Translation Learning consists of two parts:

1. Rewrite the set of training instances in $\Delta S$ form so that Rotation Learning is taken into account

2. For each of the 16 sense vector positions, use the training instances to build a discrimination tree which categorizes the sense change based on the initial sense input of a subsection of the sonar ring.

The end result will be a forest of discrimination trees, one for every (action, sense component) pair. When put together, these discrimination trees will predict the value of $\Delta S$ based on the value of $S_0$.

A Definition for $\Delta S$

It was stated before that for the sake of increased concept generality it is preferable to write each action instance in the form $(A, S_0) \rightarrow \Delta S$ instead of $(A, S_0) \rightarrow S_f$, but we have yet to define $\Delta S$. Intuitively, it is some
characterization of the difference between two sense vectors—in this case $S_0$ and $S_f$.

Given this there are still a number of possible ways to specify $\Delta S$. The most straightforward way would be to define it as a vector of the same length as $S$ such that

$$\Delta S[i] = S_f[i] - S_0[i]$$

But this characterization does not allow us to take advantage of the knowledge gained through Rotation Learning. It lumps together the sense change caused by a shift in orientation with that caused by displacement. If we assume, however, that the rotation component of an action is known then it is more effective to write $\Delta S$ as a vector of pairs defined as follows:

$$\Delta S[i] = (j, S_f[i] - S_0[j])$$

where $j$ refers to the component of $S_0$ which most nearly corresponds to $i^{th}$ component of $S_f$, as given by Rotation Learning. Consider the example shown below where the critter executes a turn-right action:

![Diagram showing initial and final sense vectors](image)

Figure 7: The critter before and after a turn-right action

It would clearly be difficult to predict the value $S_f[0] - S_0[0]$ for the turn-right action and some initial sense impression. This value would vary widely across different instances of the turn-right action, and would depend not only on the shape of the environment directly in front of the critter before the action is taken, but also on the shape of the environment to the
critter's right side before the action (i.e. that part of the environment which the critter faces after the action is completed).

On the other hand predicting the value of \( S_f[14] - S_0[0] \) should be a relatively simple matter. This value would remain fairly constant across different instances of the turn-right action, and any variance should be determined entirely by the slope of the boundary directly in front of the critter before the action is taken. This can be identified using initial input from two or three of the forward facing sense devices. By taking the rotation component of an action into account in our \( \Delta S \) characterization, we greatly reduce the amount of information needed to learn effective translation concepts.

**An Example**

Translation Learning depends on the hypothesis that the change in sense input caused by an action can be approximately predicted from the initial sense input before the action is executed. To understand better how the change in a particular sense component depends on \( S_0 \) consider the following example. Assume the critter takes the action (3 3) moving it straight ahead and imagine we wish to predict the change which will occur in sense device \( D_4 \) as a result of this action. Clearly this will depend on the local contour of the sense input from \( D_4 \) and its neighbors as can be seen in figure 8.

![Figure 8: How will input \( D_4 \) change when the critter moves forward?](image-url)
For the critter on the far left, executing the action (3 3) would leave the input from $D_4$ unchanged, since the critter would be moving parallel to the boundary on the right. For the other two critters moving forward would change the value of $D_4$, and the sign and magnitude of the change would depend on the slope and distance of the boundary relative to the critter's orientation and position.

If we assume the boundaries of the critter's environment are continuous and well-behaved (i.e. lines or simple curves), then it should be possible to correlate the change in sense device 4 with the initial input of sense device 4 and the inputs of its immediate neighbors, sense devices 2,3,5 and 6. It is this correlation which the discrimination tree generation algorithm seeks to discover.

The same process can be applied to non-straight-ahead actions when the information given from Rotation Learning is used to compensate for the change in orientation which causes the sense image to shift across sense components as the critter rotates.

The Tree Generation Algorithm

After Rotation Learning has successfully completed on some given data set for a particular action, the data is recycled for use in Translation Learning. We assume Rotation Learning has revealed which sense device $D_j$, before the action was taken, pointed in the direction most nearly the same as $D_i$, after the action; and that each action instance has been rewritten in $(A, S_0) \rightarrow \Delta S$ form. We use the symbol $\delta S$ to refer to a vector consisting of the second element of each $\Delta S[i]$ pair.

What follows is a learning algorithm for predicting $\delta S[i]$ given the action $A_k$, the initial sense vector $S_0$, and the value of $j$ from Rotation Learning. At this point we introduce the following notational simplification for the value of $S_0[j]$ and the relative values of the four nearest neighbors:
\[
J = S_0[j] \\
R = S_0[j + 1 \mod 16] - J \\
L = J - S_0[j - 1 \mod 16] \\
RR = S_0[j + 2 \mod 16] - R \\
LL = L - S_0[j - 2 \mod 16]
\]

The discrimination tree is grown using some very simple clustering techniques [Michalski 83] and again making use of the attribute correlation formula from Quinlan's work. Briefly the algorithm works as follows:

1. Given a set of training instances for a particular action and a particular sense component \( i \), if one of the following conditions holds then stop:

   (a) The number of training instances is less than some constant \( N \), and the range of the values of \( \delta S[i] \) is within some specified tolerance \( T_1 \).

   (b) The number of training instances is greater than or equal to \( N \), and a linear function of the form \( c_1 J + c_2 R + c_3 L + c_4 = \delta S[i] \) for some set of coefficients \( c_1 \ldots c_4 \) can be found such that all the training instances satisfy this function to within some specified tolerance which is equal to the standard deviation of \( \delta S[i] \) multiplied by \( T_2 \).

2. Cluster the \( \delta S[i] \) values into two disjoint sets. The clustering criterion are:

   (a) Maximize inter-cluster difference

   (b) Minimize the difference between the size of the range of the two clusters.
3. Choose an attribute \((J, R, L, RR, \text{ or } LL)\) which most nearly splits the training set into the two \(\delta S[i]\) clusters. In case of a tie between two or more attributes, choose the attribute which has the greatest range of values in the training set.

(a) Find the median value of the attribute in each of the two \(\delta S[i]\) clusters.

(b) Reclassify the training set based on the value of the attribute, by using the two medians as seeds and maximizing inter-cluster difference.

(c) Use Quinlan’s attribute evaluation function to evaluate the attribute based on how closely the attribute cluster correlates with the \(\delta S\) cluster.

4. Once the best attribute has been found, partition the training set into that attribute’s clusters.

5. Apply the entire procedure recursively to each sub-training set.

When finished, this algorithm will produce a discrimination tree where the concept at each leaf is either a linear function returning an approximate value of \(\delta S[i]\) or a set of length less than \(N\) of \(\delta S[i]\) values, which may be averaged to obtain a single value for \(\delta S[i]\). The strategy of this algorithm is to use lines and points to approximate the actual (but unknown) non-linear and possibly discontinuous function relating \(J, R, \text{ and } L\) to \(\delta S[i]\).
Results

On the following page, a tree which was generated from 120 instances of
the action (3 3) and sense component 0 (which faces directly forward) is
reproduced. The tree is described in list form, with each interior node having
the format:

((attribute value1 value2) left-descendants right-descendants)

and each leaf node having one of two formats:

((attribute value1 value2) coefficients number-of-members)

or

((attribute value1 value2) 'TS list-of-members)

The (attribute value1 value2) triple is a criterion for membership
in that node. It states that all data items belonging to this node or one of
its descendants is closer to value1 of the attribute, than it is to value2.
The coefficients are a list of 4 real numbers, (c1 c2 c3 c4), as given in the
algorithm description. If a node has less than 6 members then it is designated
as too small (TS) to use linear interpolation on, and instead a list of δS[i]
values is kept.
Translation Learning Result

(Root ((J 220 128))
  ((J 220 204))
  ((J 220 216))
  ((J 220 218))
  ((J 220 219))
  ((L 23 2))
    ((L 23 78) (0.0 0.0 0.0 0.0) 6)
    ((L 78 23) TS (-11)))
  ((L 2 23))
    ((L 0 2) TS (0 0))
    ((L 2 0) TS (-24))))
  ((J 219 220) TS (-24)))
  ((J 218 220) TS (-25)))
  ((J 216 220) TS (-25 -24 -25 -25)))
  ((J 204 220))
  ((J 185 195) (-24.1 -0.0045 -0.0015 0.0156) 7)
  ((J 195 185) (-19.5 -0.0279 0.00493 -0.00461) 10))
  ((J 128 220))
  ((J 98 111))
  ((L -16 -4))
    ((R -2 -13) TS (-26 -24 -25 -24 -26))
    ((R -13 -2) (-25.6 0.00983 -0.00780 0.0450) 10))
  ((L -4 -16))
  ((J 98 78))
    ((LL 2 -7) TS (-24 -26 -25))
    ((LL -7 2) TS (-25 -26 -25 -25 -25)))
  ((J 78 98) (-24.91 -0.00455 -0.0299 0.00763) 16)))
  ((J 111 98))
  ((R -78 -1) TS (-25 -25 -26 -24))
  ((R -1 -78))
  ((R 12 0))
    ((R 12 14) (-21.2 3.81e-4 -0.0187 -0.421) 6)
    ((R 14 12) (-23.1 -0.0161 0.0127 0.00856) 10))
  ((R 0 12) (-24.6 8.00293 -0.006 0.004742 28))))
Figure 9: A partial picture of the Translation Learning result
The tree diagram on the previous page illustrates the first 11 lines of the Translation Learning result. When the critter executes the (3 3) action in some environment the forward facing sensor \( (D_0) \) will normally decrease by approximately 25 pixels. The exception to this rule occurs when there is no boundary within range of \( D_0 \) \( (S_0[0] = 220) \) in which case the change is difficult to predict from the initial sense data, though most often no change will occur at all \( (\delta S[0] = 0) \).

The tree diagram reveals how the critter understands this situation. The data set is partitioned so that all of the instances where the value of \( J \) \( (S_0[0]) \) equals 220 are grouped together (represented by the box labeled with \( (J \ 220 \ 219) \)). This data is then further partitioned using the \( L \) attribute (remember that \( L = S_0[0] - S_0[15] \)), to isolate those anomalous cases where \( \delta S[0] \) does not equal 0. The final result is a tree which predicts that \( D_0 \) will show no change as a result of taking an the action \( (3 \ 3) \) if and only if \( J = 0 \) and \( L = 0 \) or \( 16 < L < 51 \).

As it turns out, the value of \( L \) may not be particularly relevant to deciding the value of \( \delta S'[0] \). Ideally it might have been better if the algorithm had stopped discriminating at the \( (J \ 220 \ 219) \) node and simply ignored the two anomalous cases. Still, even though the tree is more complicated than seems necessary, its accuracy is not significantly decreased because of the irrelevant nodes. The problem of over-specialization in discrimination tree generation will be discussed further in the next section.

### Analysis of the Algorithm

On the whole, as will be seen in the chapter describing test results, this algorithm performed reasonably well, with a time complexity \( (O(n \log n), n \) being the size of the training set) that is not prohibitively expensive even for large training sets. Still, the discrimination tree produced is by no means an optimal representation of all the knowledge contained in a given training set. What follows is an analysis of the interesting features of the algorithm,
along with criticism and justification for the choices which were made, and a few remarks about alternative approaches.

The Importance of Linear Approximation

The most significant feature of the discrimination trees which are produced by Translation Learning is the use of a linear approximation as a concept to describe the set of data points contained at a leaf. This greatly reduces the number of total leaves needed when compared to the strategy of allowing only one $S[i]$ value at every leaf, and gives increased accuracy when compared to the strategy of allowing a range of $S[i]$ values in any terminal node.

Of course, theoretically even greater accuracy could be obtained using fewer leaves if the data points could be approximated by other functions besides lines. In fact if one were to sit down and do a detailed trigonometric analysis using the angular difference between sense input devices one could probably find an optimum function to use for this particular sonar critter. But this violates the tabula rasa spirit of critter learning more than really seems necessary, and would severely limit the applicability of the algorithm to other sensory arrangements.

The Need for $LL$ and $RR$

It may at first seem strange that whereas linear approximation uses only the values of three sense components, $J$, $R$, and $L$, the discrimination algorithm makes additional use of $LL$ and $RR$ as possible attributes. The reason for this is that in most environments, the critter has to deal not just with continuous lines or curves, but also with line intersections, or corners. Consider the example shown in figure 10.

In the picture on the left, the critter has no way to distinguish between the two possible environments which are shown using only three sensors. Only when it considers input from the $LL$ and $RR$ sensors, as shown on
Figure 10: Two corners which are indistinguishable using only 3 sensors.

The Role of Conceptual Clustering

The use of conceptual clustering in this algorithm is an expedience. There is no guarantee that the data set partition which the clustering algorithm produces is actually the best split (using Quinlan’s measure) upon which to choose a discriminating attribute. In fact, the optimal split is usually different for every attribute. Ideally it would be better to evaluate each attribute on all possible splits and take the maximum value. This would, however, increase the computational complexity of the algorithm prohibitively.

The clustering criteria used by the algorithm are very simplistic. Assume for example that our ordered set of $\delta S[i]$ values is the following:

$$[0 0 0 20 25 26 26 27 29]$$

This set would be split between the 0 and the 20 so as to maximize the difference between the largest member of the small set and the smallest member of the large set. On the other hand, these $\delta S[i]$ values:

$$[1 3 5 7 10 13 16]$$
would be split between the 7 and the 10 in order to minimize the difference between the range of the two clusters (both clusters would have a range of 7). The rational behind these two clustering criteria is first to create two clusters which are as different as possible, and second to make the split evenly in order to grow a well-balanced tree. The objective is to keep the number of nodes small, which is desirable in order to have as much data as possible at every leaf node.

**Problems with Discrimination Trees**

There are several drawbacks to the discrimination tree approach to Learning by Examples which the above algorithm does not entirely evade. The first and most fundamental problem faced by a tree generation algorithms such as Quinlan's is the question of when to stop discriminating. If we insist on discriminating to the point where each leaf contains only a single $\delta S[i]$ value, then as the size of the data at each node becomes smaller, the accuracy of the algorithm’s choice of discriminating attribute becomes less and less well-founded (and in fact when there are only two data instances to discriminate between, the choice of attribute is virtually random). On the other hand if we stop discriminating too soon, then the leaf nodes will be too general, and thus for any particular instance probably less accurate then they could be.

In the Translation Learning algorithm the degree of discrimination is determined by the constants $T_1$ and $T_2$. Though setting these constants for any particular critter is a matter of trial and error, tests did not show the error rate to be overly sensitive to changes in their value. One important feature of this algorithm is the use of the standard deviation of the data set (in conjunction with $T_2$) to decide whether or not the accuracy of the linear interpolation is acceptable. If the error tolerance were independent of the standard deviation, then actions where the variation in $\delta S[i]$ values were small, such as (1 1), would be underspecialized, while actions such as (5 5) would be overspecialized.
Another drawback to the discrimination algorithm is the ability to discriminate on only one variable at a time. Consider the following hypothetical situation illustrated in figure 11.

![Figure 11: How do we distinguish the +’s from the x’s?](image)

Assume that $X$ and $Y$ are two attributes and that we wish to separate the data points labeled +, from the data points labeled x. To do this using only one discriminating attribute at a time (i.e. with only horizontal or vertical lines) would be extremely messy and inefficient, but we could draw a single diagonal line in between the + cluster and the x cluster. Discrimination using a linear function of $X$ and $Y$, also known as multi-variable discrimination, is clearly preferable in this example.

This situation arises frequently in Translation Learning, where the value of $\delta S[i]$ is a complex function of the values of $J$, $R$, $L$, $RR$, and $LL$. Using only single variable discrimination often causes unnecessary splitting of the data set, which leads to overly specialized, inaccurate concepts.
Possible Improvements

There has been research into possible ways of dealing with some of the difficulties described above. Quinlan and others have proposed a solution to over-specialization problem called tree pruning. In tree pruning a second training set is used to verify the efficacy of the leaf nodes, and so that nodes based on erroneous discriminations can be removed from the tree [Quinlan2 86]. This scheme has the disadvantage of requiring two training sets where only one was needed before, so that less data is available for use in tree generation. Pessimistic Pruning, proposed by Quinlan, is an algorithm for pruning without the use of a second training set, but experience has shown this algorithm to be ineffective in domains of complexity comparable to that of Translation Learning.

Finding an efficient algorithm for doing multi-variable discrimination is a difficult, and as yet unsolved problem. There has been some success with an algorithm that hypothesizes a linear splitting function and then does a hill climbing search to find the optimum coefficients [Breiman 84]. This algorithm might have potential application for Translation Learning, though its computational complexity could be prohibitive.
Chapter 7

Imaginary Actions

At this point it might enhance the reader’s understanding if we summarize what has been accomplished so far. Figure 12 illustrates the flow of information through the Action Learning algorithm.

![Diagram of the Action Learning algorithm]

Figure 12: Data flow in the Action Learning algorithm
The raw sense data collected in the training environment is used in Rotation Learning to produce a set of \((i, j)\) pairs which describe the orientation change caused by each action. These pairs are then used in conjunction with the raw sense data to accomplish Translation Learning. This produces a forest of discrimination trees (one for each sense component and action) which can in turn be used to create a \textit{predict} function. This function takes as input an action and an initial sense vector. It selects the subset of 16 trees (one for each sense vector component) which correspond to the given action, and then chooses the node from each tree which most nearly matches the appropriate portion of the given vector. Each of these nodes provides a simple equation for computing the value of a single component of the \(\delta S\) vector. By adding \(\delta S\) and \(S_0\) a prediction for the final sense impression \(S_f\) is then produced.

The remarkable aspect of the \textit{predict} function is that it can produce reasonable forecasts of action results from initial sense vectors which are different from any contained in the training set. This is due primarily to the fact that the change in each sense component is predicted using only a small portion of the entire sense vector. As long as this small portion (or something similar) can be found in some example vector from the training set, the rest of the example vector need not be the same, or even similar.

Rotation and Translation Learning discover concepts which describe the way in which sense input changes for a given action and environmental situation. But in order to act effectively on most forms of exploratory motivation, another sort of knowledge is needed. The critter must have some way of understanding what actions do relative to each other in terms of position change independent of a particular environment. In order to act on the impulse, “Explore!” the critter needs to know which actions actually cause it to change its position in the environment (as opposed to merely rotating in place), and in order to act on the impulse, “Go fast!” the critter needs to have some means of comparing action magnitudes.

To acquire such knowledge we assume that we are given the following
useful piece of information, a priori: the overall sense change caused by taking an action in a uniform sensory situation is proportional to the displacement magnitude of the action. The difficulty, of course, is that the critter may not have encountered any such uniform sensory situations in its training environment. Nevertheless, using the predict function, the critter can create such situations artificially, in effect imagining what would happen in a uniform sensory situation if the action were taken.

The following procedure ranks critter actions according to relative displacement:

1. Using the knowledge gained through Rotation and Translation Learning, predict the result of taking each of the 27 different actions from a set of uniform sense situations (i.e. those where all the components of $S_0$ have the same value, as if the critter were positioned in the center of a circle of arbitrary radius).

2. For a given action $A_k$ and some uniform $S_0$:

   $$DIS_k = \sum_{i=1}^{16} |\delta S[i]|$$

3. The relative displacement value for a particular action $A_k$ is the sum of the $DIS_k$ values over the given set of imaginary uniform sense impressions.

4. Consider those actions which have a relative displacement value of approximately 0 (within some specified tolerance) to be rotation actions.

The end result is a partition of critter actions into two groups: (1) pure rotations and (2) translations. The translation actions are ranked according to their relative displacement values. An example of this ranking for the sonar critter after 180 training instances per action is shown on the next page:
<table>
<thead>
<tr>
<th>Translations:</th>
<th>Rotations:</th>
</tr>
</thead>
<tbody>
<tr>
<td>(5 5): 5927</td>
<td>(-5 5): 170</td>
</tr>
<tr>
<td>(4 5): 4905</td>
<td>(3 -3): 97</td>
</tr>
<tr>
<td>(3 4): 4867</td>
<td>(5 -5): 81</td>
</tr>
<tr>
<td>(5 4): 4831</td>
<td>(-4 4): 74</td>
</tr>
<tr>
<td>(5 3): 4787</td>
<td>(-2 2): 72</td>
</tr>
<tr>
<td>(4 3): 4739</td>
<td>(2 -2): 67</td>
</tr>
<tr>
<td>(3 5): 4680</td>
<td>(-3 3): 64</td>
</tr>
<tr>
<td>(2 5): 4336</td>
<td>(4 -4): 58</td>
</tr>
<tr>
<td>(5 2): 4209</td>
<td></td>
</tr>
<tr>
<td>(4 4): 3865</td>
<td></td>
</tr>
<tr>
<td>(2 4): 3682</td>
<td></td>
</tr>
<tr>
<td>(3 3): 3579</td>
<td></td>
</tr>
<tr>
<td>(4 2): 3538</td>
<td></td>
</tr>
<tr>
<td>(1 2): 2624</td>
<td></td>
</tr>
<tr>
<td>(2 2): 2476</td>
<td></td>
</tr>
<tr>
<td>(3 2): 2450</td>
<td></td>
</tr>
<tr>
<td>(2 1): 2434</td>
<td></td>
</tr>
<tr>
<td>(2 3): 2406</td>
<td></td>
</tr>
<tr>
<td>(1 1): 1287</td>
<td></td>
</tr>
</tbody>
</table>

Clearly this action ranking is not perfect. The values for relative action displacement are only accurate to ±200 pixels, and as a result some actions have been inverted (one example being (3 5) and (4 3)). Nevertheless the goal of finding an approximate, qualitative way of comparing action magnitudes and of separating rotations from translations has been met.

The application of Imaginary Actions is not properly a part of sensory-motor coordination or of Action Learning. Its use is primarily an expedient for gaining knowledge about the relative magnitudes of actions—knowledge
which is essential for demonstrating the effectiveness of Action Learning concepts in the exploration of unfamiliar environments.
Chapter 8

Testing

Before learning begins, the critter gathers data within a training environment. The choice of actions inside the training environment is random. During training, the critter has a built-in reflex mechanism which keeps it from coming too close to any boundary—halting the selected action and turning the critter around before a new action is taken. Such instinctive self-preservation mechanisms (like the reflex actions of newborn infants who push their head against the back of a seat when confronted by a looming object [Bower 77]) are not necessarily incongruent with the spirit of tabula rasa learning, as will be discussed in Chapter 10.

After running the algorithm on the data gathered from the training environments, two kinds of tests were conducted to prove the effectiveness of the Action Learning algorithm for the sonar critter. The quantitative tests show how the predictive accuracy of the critter increased with greater learning, and also that the knowledge gained in a training environment is applicable to different, unseen environments. The qualitative tests verify that the accuracy of the Action Learning concepts is sufficient to allow the critter to achieve a sensory goal state, given some reasonable measure of sense vector goodness.
Quantitative Tests

In quantitative testing we evaluate the critter's ability to guess the effect of an action given the initial sense impression, based on the knowledge it has learned in training environments. The learning algorithm was run on three different training sets: the first composed of 60 instances of each of the 27 different actions (1620 total instances) taken from random positions inside a rectangle of size $270 \times 300$ pixels, the second composed of 120 instances—the same 60 that were taken from the rectangle plus 60 more from random positions inside a circle of radius 125 pixels, and the third composed of 180 instances—the same 120 from the second training set, plus 60 more from within a curved corridor. The knowledge learned from these three training sets was then tested by predicting the effect of actions taken in different positions inside the rectangle, circle, corridor, and four other environments illustrated in figure 14.

The results of these tests are shown below. For comparison the average error of a stupid critter, which always predicts that $S_f = S_0$ is also shown.

<table>
<thead>
<tr>
<th>Test Environment</th>
<th>Stupid Critter</th>
<th>60 Rectangle</th>
<th>60 Rectangle + 60 Circle</th>
<th>60 Rectangle + 60 Circle + 60 Corridor</th>
</tr>
</thead>
<tbody>
<tr>
<td>rectangle</td>
<td>32.5</td>
<td>4.3</td>
<td>4.4</td>
<td>4.0</td>
</tr>
<tr>
<td>circle</td>
<td>29.3</td>
<td>5.1</td>
<td>1.7</td>
<td>1.7</td>
</tr>
<tr>
<td>corridor</td>
<td>42.0</td>
<td>5.7</td>
<td>5.2</td>
<td>4.6</td>
</tr>
<tr>
<td>triangle</td>
<td>34.1</td>
<td>4.0</td>
<td>4.0</td>
<td>3.6</td>
</tr>
<tr>
<td>oval</td>
<td>37.3</td>
<td>4.9</td>
<td>3.0</td>
<td>3.1</td>
</tr>
<tr>
<td>weird shape</td>
<td>32.1</td>
<td>4.7</td>
<td>2.9</td>
<td>2.5</td>
</tr>
<tr>
<td>polygon</td>
<td>38.9</td>
<td>4.5</td>
<td>4.2</td>
<td>3.8</td>
</tr>
</tbody>
</table>

Figure 13: Average Error per Sense Component (in number of pixels)

These results show that the knowledge the critter gains in a training
environment can be successfully applied to unexplored environments. We see this especially in the weird shape and polygon, where the error decreased significantly after learning in each environment. This indicates that the strategy of learning concepts which apply to small sections of the sonar ring was effective in allowing the knowledge acquired in one environment to be applied in very different surroundings. Also notice that learning in the circle and corridor did not adversely affect the knowledge gained in the rectangle.

Qualitative Tests

The purpose of qualitative testing is to show that the concepts acquired through action learning may be used to achieve a well-defined sensory goal. A well-defined sensory goal is one for which a numerical evaluation of the relative "goodness" of any given sense vector exists. We examine critter behavior on two such sensory goals, which are referred to as Find-Center, and Follow-Right-Wall.

The Claustrophobic Critter

The Find-Center sensory goal is defined using the following very simple evaluation function:

$$FC(S_0) = \min_{i=0}^{16} S_0[i]$$

Thus to maximize such a function the critter must select actions which make the distance from the nearest boundary as great as possible (hence the term "claustrophobic"). In the tests the critter selected actions based on the following strategy:

1. Predict the result of taking any of the 27 different actions from the current situation, $S_0$. Let $\text{predict}(A, S_0)$ stand for this prediction for some action, $A$. 
2. If the value of $FC(S_0)$ is greater than or almost equal to the value for all 27 actions of $FC(predict(A, S_0))$, then goto (3). Otherwise execute the action which maximizes $FC(predict(A, S_0))$.

3. For each action $A_k$ for which $FC(predict(A_k, S_0))$ is within some tolerance of $FC(S_0)$, call the function $predict(A_j, predict(A_k, S_0))$ instantiating $A_j$ with all 27 possible actions.

4. If $FC(S_0) \geq FC(predict(A_j, predict(A_k, S_0)))$ for all $A_k$, $A_j$ combinations then quit. Otherwise execute the action $A_k$ which maximizes the $FC$ function.

This strategy insures that the critter selects actions which immediately move it closer to the center of a symmetrical environment whenever possible. If no such action is predicted to do this, then the critter looks to see if any combination of 2 actions will move it significantly closer (i.e. if the critter faced directly away from the center of the environment it would have to first turn around and then move forward). The critter halts when no two actions are predicted to increase significantly the distance from the nearest wall.

In figure 15 the resulting traces from typical trials of the Find-Center test are shown. In all of these trials the critter used knowledge gained from learning only inside the rectangle (training set 1). The critter always began facing away from the center and, its first action was always a rotation (though this cannot be seen in the figure). The other actions are represented by the lines in between the dots on the diagrams. In each case the critter halted in a spot within 5 pixels of an optimum position in the environment.

**The Exploratory Critter**

The Follow-Right-Wall sensory goal is defined by the function:

To maximize this function the critter must select actions which keep the right hand boundary (that boundary in the direction of sensors 3, 4, and 5) as near as possible to 50 pixels distant. In the tests the critter was given knowledge learned from the third training set, and selected actions based on the following strategy:

1. Using the results from taking Imaginary Actions, partition the set of possible actions into rotations and translations.

2. Select a rotation action whose resulting sense impression maximizes the sum of senses 15, 0, and 1 (in other words, turn so as to face the most open space). Execute this action.

3. Find a set of translation actions which are predicted to maximize the FRW function (to within some tolerance), from among those translation actions which are not expected to cause any sense device to have a value less than 30 (avoid collisions with boundaries). **Stop if no safe translation action can be found.**

4. Of those actions which maximize the FRW functions, select the one which causes the greatest displacement (as indicated from taking Imaginary Actions). Execute the action. Go to (3).

Using this strategy in a corridor type environment causes the critter to first turn towards an open space, and then follow a path parallel to the right hand wall, moving as quickly as possible. Figure 16 shows the resulting traces from two typical trials of the Follow-Right-Wall test, using knowledge gained from learning in all three training environments (training set 3). Again the actions taken are represented by the lines between the dots in the diagram. Though the critter’s limited range of actions did not allow it to follow the wall exactly, the distance from the right hand wall remained fairly constant throughout the critter’s journey.
Analysis

We believe that the results of our Qualitative Tests show that the Action Learning critter is capable of acquiring knowledge in a training environment which can be applied to unexplored environments. The tests show that the critter can find "interesting" places in unknown environments and explore in between such places safely and intelligently. We have thus successfully accomplished the primary goals of Action Learning.
Figure 14: The Critter Environments
Figure 15: Results of Find-Center in Three Environments
Figure 16: Results of Follow-Right-Wall Tests
Chapter 9

Conclusions

This thesis has described one general approach to the learning of sensory-motor coordination in exploratory robots. The key idea in this approach is to separate the sense change caused by the rotation component of an action from that caused by the translation component. This allows action learning to be done at the sense vector component level instead of the sense vector level, which facilitates more general concepts.

Though the specific implementation described here for the sonar critter may not be applicable to all sensory motor systems, we claim that the strategy of separating sensory image rotation from image translation should be generally applicable to a wide range of critters. In fact it seems probable that any robot which must navigate in some unknown environment using information provided by directional sensors could profit from the following learning strategy:

- Relate the orientation change caused by a particular action to the directional differences between sense inputs (Rotation Learning).
- Develop concepts to describe sense changes for a particular action and sense component using only a small number of local initial sense inputs (Translation Learning).
- Develop a qualitative understanding of relative action displacement (Imaginary Actions).
We have presented the implementation of this Action Learning strategy for the sonar critter as strong evidence for its ultimate applicability to practical, mobile robot sensory-motor systems. We have achieved our original goal of learning from examples the sensory change caused by an action, without using or creating a map of the environment—deriving sensory-motor concepts which may be used in unexplored environments.

Two criticisms which might be leveled against our particular implementation of Action Learning are: (1) that it tailors too much towards the particular sense devices and arrangement of the sonar critter, and (2) that so much \textit{a priori} knowledge is given to the critter that the spirit of \textit{tabula rasa} learning has been lost.

In answer to the first criticism we would argue that as far as Rotation and Translation Learning are concerned, there is nothing inherent in either of these algorithms which could not apply (with some slight modification to tolerance factors, etc.) equally well to any directional sensory system where the possible sense inputs can be meaningfully mapped onto a finite range of integers, as for instance the gray-scale inputs of video camera pixels mentioned in Chapter 2. In support of this claim we would note that neither the Rotation Learning algorithm (which uses Quinlan's information theoretical approach to choose attributes), nor Translation Learning (which uses a linear approximation to describe concepts) are tailored to the sonar critter. Both of these algorithms rely exclusively on the numerical and statistical relationships of the data points to form concepts. Obviously the most incontrovertible way to answer the first criticism would be to show the algorithm effective for several different types of critters, but this is a non-trivial, time consuming task which unfortunately must be left for future research. The second criticism will be addressed at length in the next chapter.
Chapter 10

Much Ado about Nothing

The phrase *tabula rasa* learning has been mentioned rather glibly throughout this thesis, but what does it really mean to learn something from nothing? Drew McDermott, in describing the problems faced by research on a General Purpose Learning Mechanism wrote that, “In order for an organism to learn anything it must already know a lot. Without strong clues to what is to be learned, nothing will get learned.” If this is true then it would seem to sound a death knell for all *tabula rasa* learning, which by its very definition attempts to learn without knowing anything at all.

Yet even if one accepts McDermott’s statement as a universal truth, this does not necessarily imply that there is no hope for *tabula rasa* learning. The reason for this apparent contradiction stems from a misunderstanding about what *tabula rasa* critter learning actually is—which itself originates from our imprecise views about what constitutes knowledge.

Some Questions about Knowledge

By now it should be clear to anyone who has read this far, that the Action Learning Critter does not begin to learn in an intellectual vacuum. The very structure of the critter’s learning algorithm assumes a great deal about the types of sense input which will be received and motor actions which
can be performed. But do these structural assumptions really constitute knowledge? Not unless a caterpillar knows that wrapping itself in a cocoon and going to sleep will cause it to change into a butterfly; or a tree knows that shedding its leaves in fall is more efficient than keeping them through the winter months. Clearly the vast majority of organic behavior is not based upon knowledge—at least not in our ordinary sense of the word. Why then should the a priori structural assumptions of the Action Learning Critter be called by that name?

The point is that the term tabula rasa should not preclude any sort of a priori information. Clearly, if we place a critter, organic or artificial, in some environment with a completely empty head, it will just sit there until it dies or is turned off. At the very least, some sort of motivation is needed if anything interesting is to happen. In animals these motivations are instinctive. They are acquired through millions of years of natural selection, and account for the vast majority of natural behavior. It seems rather pedantic to classify such innate motivations as some sort of inherent knowledge. Primitive motivations and instincts are structural properties, like arms, legs, and sonar devices, which are present from birth till death. They are in no way inconsistent, and in fact are absolutely essential, to the notion of tabula rasa learning.

How then do we define knowledge? In animals we might be able to get away with an answer like: “Knowledge is simply what has been learned.” Using this definition every animal begins at birth with no knowledge. Some species, such as insects, live out their lives without learning (or knowing) anything at all, while other species of animals acquire knowledge through experience. By this definition everything an animal knows was once learned, and thus at birth then every animal is a tabula rasa critter.

But whether or not one accepts this definition for animals, the question of What is knowledge? cannot be decided so easily for artificial critters. We cannot (yet) enlighten a dog in the art of fetching a stick by surgically altering its brain, but we can tell a robot how to perform such a task by
programming it. Suppose one robot learns how to fetch a stick through trial and error using some learning algorithm, and then passes this information onto a second robot via a memory dump. Clearly we cannot claim that the first robot knows (because it learned) how to fetch a stick, while the second robot only does so instinctively (because it did not learn). Unless we accept programming as a form of learning (learning by brain surgery?) we have to find another definition for knowledge. Or stated another way, we need to decide how smart we will allow a tabula rasa critter to be, before we no longer say that it, “knows nothing”.

One way to help resolve this question is to observe the decisions which natural selection has made in designing nature’s most sophisticated tabula rasa critter—the human infant. In the following sections we examine the differing types of a priori information which may be provided to artificial critters and judge whether or not such information should be considered a violation of the tabula rasa condition, using the human critter as our standard for comparison.

Motivational Systems

As was stated before, we do not feel that primitive motivation should be classified as knowledge. In animals, “brains have evolved systems which transform information on bodily needs and environmental events into cerebral activity producing either pleasure (comfort) or pain (discomfort). These systems are known to psychologists as motivational systems …” [Aubin 88]. This “information on bodily needs and environmental events,” basically boils down to sense data. Every critter, whether real or artificial needs to be given a priori not only a motivational system, but also a way of interpreting its senses so as to tell whether or not its motivations have been satisfied.

In the sonar critter this corresponds to the control strategies used in Qualitative Testing. The goal of holding three particular sense components at a constant value in the Follow-Right-Wall experiment is an example of a
direct link between motivation and the sensory system. This is somewhat analogous to the instinct in humans to maintain the equilibrium of fluid levels in the inner ear which makes us feel uncomfortable whenever we start to lose our balance. This sort of motivational link to the sensory system should not be classified as knowledge. It is in fact a prerequisite for intelligent, goal-directed behavior in both natural and artificial critters.

**Knowledge about Actions**

In addition to the strong link between the senses and the motivational system, there is also a direct instinctive link between the motivational system and some motor activity in animals. Human infants exhibit several instinctive movement reflexes immediately after birth (e.g. Tonic Neck Reflex and Rooting Reflex) which help insure their survival before learning begins [Bushnell 81].

The analogy to instinctive motor activity in the sonar critter would be the random actions which are taken in the training environment in order to gather data for Action Learning, and the reflex to avoid walls while collecting this data. Again, this pre-programmed behavior is not knowledge, nor is it incompatible with the spirit of *tabula rasa* learning. In the early stages of mental development critter survival depends on instinctive curiosity and self-preservation reactions.

In his book, *Vehicles*, Valentino Braitenberg hypothesizes a design for a simple mechanical mechanism which would exhibit the behavior of turning toward or away from a particular type of sense input—much like the critter’s reflex to avoid walls. Braitenberg goes on to suggest that such direct connections between sensors and motors are the first step towards thinking machines. The development of this sort of reflex in nature may have been one of the primary factors which led to the evolution of intelligent life on this planet [Braitenberg 84].

Still, one must draw the line when it comes to giving a robot *a priori*
information about the effect which taking a particular action will have on its sense input. Humans and other higher animals begin to learn such information soon after birth, and continue throughout their lives to have the ability to adapt to sensory distortion or changes in motor reaction [Lackner 81]. Acquiring sensory-motor coordination should be a fundamental part of tabula rasa learning in robots, just as it is in humans.

Interrelation of Senses

The problem of how human infants coordinate the information they receive from different sense modalities has puzzled cognitive psychologists for some time. For instance, how does an infant come to know that when it hears its mother’s voice off to the right, it will be able to see its mother by turning its head to the right. Research seems to indicate that “at least a crude form of intersensory integration is present from early infancy,” but also that performance definitely improves with age [Abravanel 81].

In the sonar critter this is analogous to the a priori assumptions about each sense device having the same form and function, as well as the information given about the neighbor relationships between sense devices. In the absence of any firm philosophical or psychological evidence as to whether such assumptions constitute knowledge, and thus violate the spirit of tabula rasa learning, one is inclined to take a purely pragmatic view of the matter. Assuming the critter’s brain is not going to be suddenly transplanted from one sensory system to another which is totally different, and assuming that all sense devices function properly throughout the critter’s life span, then there is no question of adaptation involved—which is always one of the primary practical justifications for learning in both robots and animals. It would also be difficult to conceive of a sensory system which would require an unmanageable amount of programming in order to describe in general terms the spatial and functional relationships between sense devices. There seems to be no real practical benefit to be derived from the application of learning
in this area—a fact which nature, perhaps, has already discovered.

**Generality vs. Efficiency**

The type of *a priori* information which is the most difficult to pin down in terms of whether or not it represents knowledge or instinct, is the information contained in the very structure of the learning algorithm. An example of this type has already been discussed in Translation Learning where it was argued that giving the critter a very specific sort of trigonometric function with which to interpolate data points would violate the spirit of *tabula rasa* learning. But in what sense can we justify calling this sort of *a priori* information, knowledge?

Probably the most sensible way in which to classify this sort of information is by the following simple rule: whenever we render a learning algorithm less general in order to make it more applicable to a particular situation, then we have added knowledge to the algorithm. Knowledge which might have been gained through learning, but has instead been given *a priori* via the structure of the algorithm. The key element here is the loss of generality in favor of specialization. Whenever we limit the applicability of our algorithm in order to gain efficiency, we are to one degree or another in violation of the *tabula rasa* criterion.

Of course, the Action Learning Critter described in this paper does make such assumptions. In fact, all of the assumptions and restrictions listed in Chapter 3 decrease the generality of the algorithm to some extent. My justification for this would be that in order to make a practical learning algorithm we have to limit its applicability to a domain of reasonable size and complexity. Just as nature specializes the brain of each animal to be appropriate to that animals abilities and needs, so the computer scientist must tailor the critter learning algorithm to the robot's sensory-motor capabilities. Always keeping in mind, of course, that the more *a priori* knowledge which is given, the less generally applicable will be the algorithm.
The conclusion which may be drawn from all of this is that the *tabula rasa* critter is in no way equivalent to the General Purpose Learning Mechanism, disparaged by McDermott. Natural critters, including human infants, are not general purpose learners, but quite the contrary are limited and focussed by a large amount of *a priori* information given in the form of instinct. The new-born infant mind is like a dry river bed into which knowledge flows, and the structure of the infant mind predetermines to a great extent what sort of knowledge can be learned, as well as facilitating the learning task.

Artificial critters are no different in this. The structure of their learning algorithms act in the same way that instinct does in nature—to focus attention on those aspects of sensory information which can most readily be organized in a coherent fashion for absorption as knowledge. Without this instinct no critter, either natural or artificial, would have a reasonable chance to learn about its environment.
Chapter 11

Areas for Future Research

The work described in this thesis is only the first step towards a practical Action Learning critter—one which begins by knowing virtually nothing, and ends up exploring a real environment safely and intelligently. This chapter describes the different areas where work still needs to be done before such a critter can be developed. This is by no means a comprehensive description of all unsolved problems in critter research, but covers just those areas which were explored in the course of this thesis work but not pursued for lack of time or interest.

Eliminating A Priori Assumptions

One area for further development which has already been touched upon is the elimination of those a priori assumptions given in Chapter 3. A list of some of these assumptions, along with some suggestions as to how they might be eliminated is given below:

- Range of Possible Sense Inputs: This can be learned by simple observation, taking note of the greatest and smallest sense values to be found in the training set.
- Neighbor Relation Between Sense Devices: A very simple algorithm for finding the nearest neighbors of a sense device for the sonar critter is
the following:

1. Consider a training set of sense vectors taken from random positions in some arbitrary environment, and assume one wants to know the nearest neighbors to sense device \( D_i \).

2. For each sense device \( D_j, j \neq i \), sum the values over all the sense vectors of \( |S[j] - S[i]| \).

3. The nearest neighbors will be the sense devices which have the smallest summations.

This algorithm assumes that in most situations the sense input from one sense devices will be similar to that of its nearest neighbor(s), generally a little greater or a little smaller, but usually almost the same. Thus finding the sense device(s) with the smallest absolute difference to a given sense device gives a strong indication that the sense devices are neighbors. This algorithm worked very well in tests done on LASI, though it may be difficult to justify generalizing it to other sensory systems.

- Error Tolerance: Since the predictions given for sense change by the Action Learning algorithm are only approximations, small errors (on the order of 5%) in sense device input probably would not cause a significant degradation of performance. Handling errors of larger magnitude, either from the sense devices themselves or from inconsistent effects of actions, might require some modification to the Translation Learning algorithm. Tree Pruning techniques [Quinlan2 86] have been shown to be helpful in discovering and eliminating those nodes of a discrimination tree which contain mostly bad data.
Qualitative Understanding of Actions

In order to be able to reason intelligently about actions, the critter needs to know more than just their approximate effect on sensory input. There also needs to have some sort of qualitative knowledge about the positional changes which result from movement. The use of Imaginary Actions is only a partial solution to this problem. The critter should also which actions are inverses of each other (e.g. (-3 3) and (3 -3)) or which sets of actions are equivalent (e.g. (2 2) + (3 3) = (5 5)).

One possible method of gaining qualitative knowledge would be for the critter to build up an Action Result Graph. The arcs of this graph would be labeled with actions and the nodes would represent action results (except for the root node which would be the starting place). An example of what a partial graph might look like is shown in figure 17.

![Figure 17: An Action Result Graph](image)

When fully completed, such a graph could help the critter avoid endless looping behavior and reduce the search space of possible actions at any given choice point. No algorithm has yet been developed for constructing such a graph, though it might be accomplished (albeit somewhat inefficiently) by a simple trial and error search routine.
Generalization over Actions

Allowing the sonar critter to have a fixed set of 27 possible actions immensely simplified Action Learning, since it allowed us to focus our attention on generalization of sense inputs. But in many real world applications it would be desirable to allow a virtually infinite set of actions (e.g. allowing any real number in a certain range to be instantiated in the \((x, y)\) action pair). In order to accomplish Action Learning for such a critter, it would be necessary to be able to do some generalization over actions as well as sense vectors.

This might be accomplished for the sonar critter by allowing the Translation Learning algorithm to have access to the action pair, \((x, y)\), and then to predict the sense change as a function of \(J, R, L, x,\) and \(y\). Thus we would not have a different tree for each individual action, but one enormous discrimination tree describing the sense change resulting from any critter action and initial position. Rotation Learning would still be done using only integral values of \(x\) and \(y\), interpolating the rotation value of non-integral actions using known values of actions which are nearly the same.

Unfortunately this algorithm would probably require much more data to learn effective action concepts than the current implementation. This is due to the fact that leaf nodes will require increased data for linear interpolation (because of the two extra variables). Also larger discrimination trees only magnify the problems discussed at the end of Chapter 6. Almost certainly, some form of practical algorithm for multi-variable discrimination would have to be discovered before one-tree Translation Learning would be feasible.
Chapter 12

Glossary of Notation

*tabula rasa* Latin for, “blank slate” Used to mean a mind before receiving outside impressions.

*A* or *A*_k  Refers to a particular critter action, named by the subscript *k*.

*D_*k  Refers to a particular sense device. The sonar critter has 16 such sense devices, numbered from 0 to 15.

DIS_ *k*  The absolute sum of the sensory change caused by the action *A*_ *k* in some initial situation.

*FC(S)_0*  A function which returns the minimum value of the vector *S*_0. The Claustrophobic Critter’s goal is to select actions which maximize this function.

*FRW(S)_0*  A function which is maximized when the critter is approximately 50 pixels from the right hand boundary.

*J*  For some sense component, *i*, and some corresponding sense component given by Rotation Learning, *j*; *J* = *S*_0[*j*], where *S*_0 is the initial sense vector of some training instance.

*L*  The difference between *J* and the value of the left neighbor of *J* for some training instance.

LASI  Learning Actions from Sonar Input. The name of the particular critter implemented in this research.

65
$LL$ The difference between $L$ and the value of the left neighbor of $L$ for some training instance.

$N$ The fewest number of training instances which may be contained in a leaf node of the Translation Learning discrimination tree and still allow linear interpolation to be done.

$\text{predict}(A, S_0)$ A function which accesses the knowledge gained by applying the Action Learning algorithm on some training set. Returns a prediction of the result of executing action $A$ given the initial sense vector $S_0$.

$R$ The difference between $J$ and the value of the right neighbor of $J$ for some training instance.

$RR$ The difference between $R$ and the value of the right neighbor of $RR$ for some training instance.

$S$ A sense vector – contains the integer input of all sense devices at a given instant in time.

$S_0$ The initial sense vector – the sense reading just before an action is taken.

$S_f$ The final sense vector – the sense reading just after an action has been taken.

$\Delta S$ A characterization of the change in sense input from $S_0$ to $S_f$.

$\delta S$ A vector of $S_f[i] - S_0[j]$ values where $i$ ranges from 0 to 15, and $j$ is given by Rotation Learning.

$simplify(S, i)$ A function which maps the value of $S(i)$ onto one of six simplified values depending on $S(l)$ and $S(r)$, the values of the left and right neighbors of sense device $D_i$.

$T_1$ and $T_2$ Two tolerance factors which determine the amount of discrimination which will be done by Translation Learning.

$(x, y)$ An energy pair – For LASI the energy pair names a particular action.
Bibliography


[Kuipers 85] Benjamin J. Kuipers, The Map Learning Critter, University of Texas Artificial Intelligence Laboratory, 1985, AITR85-17.


VITA

William Scott Spangler was born in Tampa, Florida, on March 2, 1964, son of William Henry Spangler and Marjorie Beutel Spangler. He graduated from the Massachusetts Institute of Technology in 1986 with a Bachelors degree in Mathematics with Computer Science. In June of 1986 he began work for the General Motors Advanced Engineering Staff in Warren, Michigan. In September of that same year he was granted a GM Deferred Employment Graduate Fellowship to study Computer Science at the University of Texas for 2 years.

The author's mailing address is:

Scott Spangler
General Motors Technical Center
B/MD-70
30300 Mound Rd.
Warren, MI 48090-9040

This thesis was typed by W. Scott Spangler.