A COMPUTATIONAL MODEL OF VISUO-MOTOR DEVELOPMENT

by

Anthony Passera

S.B., Mathematics
M.I.T., 1974

M.S., Computer Science
Yale University, 1981

Submitted to the Department of
Brain and Cognitive Sciences in Partial Fulfillment of
the Requirements for the Degree of

Doctor of Philosophy

at the
Massachusetts Institute of Technology
February, 1993

© Massachusetts Institute of Technology 1993
All rights reserved

Signature of Author

Department of Brain and Cognitive Sciences
November 2, 1992

Certified by

Tomaso A. Poggio

Professor, Department of Brain and Cognitive Sciences
Thesis Supervisor

Accepted by

Emilio Bizzi

Chairman, Department of Brain and Cognitive Sciences

MASSACHUSETTS INSTITUTE
OF TECHNOLOGY

NOV 13 1992

LIBRARIES
A COMPUTATIONAL MODEL OF VISUO-MOTOR DEVELOPMENT

by

Anthony Passera

Submitted to the Department of
Brain and Cognitive Sciences in Partial Fulfillment of
the Requirements for the Degree of Doctor of Philosophy

ABSTRACT

This project explores hypothetical mechanisms that may underlie visuo-motor development in humans. With an approach that combines theory and experiments, in the form of computer simulations, we studied two visuo-motor tasks: visual grasp to moving targets, and the reaching phase of tactile grasp. In these efforts, we have tried to keep our theory as consistent as possible with developmental theory and data.

The algorithm for learning articulator control uses goal-directed, self-produced, trial-and-error movement attempts toward a visible target. This is an unsupervised learning algorithm. When used with a simulated, 2-d, dynamic, robot arm, the algorithm discovers how to make straight line, minimum-jerk trajectories without being given a desired trajectory as a reference, and without the use of separate training and execution phases. The same algorithm is used for the visual-grasp task by swapping the simulated senses, motors and constraints in the form of task-specific cost functionals.

The system learns by generating, executing and evaluating control structures, or schemas (as Piaget used the term), for performing a task. New schemas are generated by modifying the more successful prior schemas, where success is measured with respect to the task-specific cost functionals. The modifications, or schema transformations, are of two general types: stochastic changes in the actions produced, and changes in the structure, especially the temporal structure, of a schema (e.g. control structure cut and paste operations, such as the crossover operation of genetic algorithms). A modified schema becomes the next one to try when an appropriate target object becomes available. If the new schema is more successful than the previous one, then the new schema becomes the reference for future search. Furthermore, the transformations responsible for the improvement are credited toward becoming part of a practise strategy for the task. What we are calling a practise strategy is a problem-specific search strategy. As the system is trying to develop more effective schemas, it simultaneously develops a practise strategy. Consequently, the system not only learns to produce the desired movement, but also learns progressively more efficient ways of training itself for similar movements toward new targets.

Thesis Supervisor: Professor Tomaso Poggio
ACKNOWLEDGMENTS

First, I am deeply grateful to my thesis advisor, Tommy Poggio. Without his guidance, support, openness, and astonishing range of interests, this thesis would simply not have happened. I would also like to thank my readers, Rachel Clifton, Ellen Hildreth, and Mike Jordan for persevering with a document that is wordy and difficult to follow.

For interesting projects and courses, I am very fortunate to have learned from Jerry Schneider, Dick Held, Arthur Lander, Mriganka Sur, Peter Schiller, Ann Graybiel and Emilio Bizzi.

I am especially grateful to Meg Turner, a dear friend, to whom understanding and healing are unquestionably more important than ambition.

For showing me why my teaching was flat, I must express profound thanks to the best teachers I know: Dorothy Austin, Annie Rogers and Meg Turner. Their teaching derives power from their courage to tell the story undisguised.

My heartfelt thanks go to Peg McAdam, for questioning everything; Sheila O'Keefe, for seeing the simplicity; Margaret Keiley, for explaining affect; Sid Blatt, for timely and fatherly encouragement; Paul Summergrad, for hearing my story; Dave Berry, for unexpected conversations; and Steve Smith, for teaching me to trust what I know.

I would also like to thank my fellow members of B&CS for many enjoyable conversations, especially Laurie Collier, Young Kwon, Jong-on Hahn, Laurie Carman, Anna Roe, Randy Smith, Diana Smetters, Lyle Borg-Graham, Bobby Dolan, Janine Mendola, Emanuela Bricolo, Alessandra Angelucci, Sherif Botros, Sonal Jhaveri, Reha Erzurumlu, Nikos Logothetis, Manny Esquerra, DaYu Woo, and Joe Bauer.

Finally, I want to thank Jan Ellertsen, who continues to make B&CS a home by watching out for graduate students...
# Table of Contents

**Introduction** .................................................................................................................. 9

**Chapter 1** Theory and metatheory for a developmental model of sensor-motor coordination ............................................................................................................. 10

1.0 Introduction .................................................................................................................. 10
1.1 The nature of knowledge .............................................................................................. 11
1.2 The process of knowing ............................................................................................... 15
1.3 The role of affect in learning ....................................................................................... 16
1.4 Learning to learn ......................................................................................................... 18
1.5 Summary ..................................................................................................................... 21

**Chapter 2** The Development of Visual Grasp and Tactile Grasp in Infants .......................................................................................................................... 23

2.0 Introductory remarks on the developmental findings for vision and motor skills .... 23
2.1 The Developmental of Reaching and Grasping in Infants ........................................... 23
   2.1.0 Chronology of developmental events for arm movements .............................. 23
   2.1.1 Reaching onset and accuracy ........................................................................... 25
       Infant studies ........................................................................................................... 25
       Hypotheses amenable to simulation (set 1) ........................................................... 26
   2.1.2 Organization of movement into units ............................................................... 26
       Infant studies ........................................................................................................... 26
       Hypotheses amenable to simulation (set 2) ........................................................... 28
       Adult studies .......................................................................................................... 28
   2.1.3 Linearity, duration, and peak velocity of reach .................................................. 29
       Infant Studies ........................................................................................................... 29
       Hypotheses amenable to simulation (set 3) ........................................................... 30
       Adult Comparison ................................................................................................... 30
   2.1.4 The curvature-speed relation and inferences about the control function .......... 30
       Infant Studies ........................................................................................................... 30
       Hypothesis amenable to simulation (set 4) ........................................................... 32
       Adult Comparison ................................................................................................... 33
   2.1.5 Relative use of visual and proprioceptive input in reaching task ....................... 34
       Infant Studies ........................................................................................................... 34
       Hypothesis amenable to simulation (set 5) ........................................................... 37
       Adult Comparison ................................................................................................... 37
   2.2 Adult reaching movements ....................................................................................... 39
   2.3 The Development of eye movements in infants ....................................................... 40
      2.3.1 Saccadic eye movements ............................................................................... 41
          Infant studies ....................................................................................................... 41
          Hypotheses amenable to simulation (set 6) ........................................................ 41
3.7.3 Storage of nearby "state" as experienced in movement attempts ........................................ 82
3.7.4 Local approximation of the cost surface ........................................................................ 86
3.7.5 GRBF interpolation of new CSEs from existing data .................................................... 89
   Choice of data points for approximation .............................................................................. 89
   Moving centers ......................................................................................................................... 89
   Choice of interpolation strategy .............................................................................................. 90
   Choice of when to interpolate and when to save the interpolant ........................................... 90
   Frequency of computing the matrix, G ..................................................................................... 91
3.7.6 Balancing practise and interpolation .............................................................................. 91
3.7.7 Summary of local database operation .............................................................................. 93

3.8 Chapter summary .................................................................................................................. 93

Chapter 4 The Variation Level of the Interactive Architecture ................................................. 94

4.0 Introduction ............................................................................................................................. 94
4.1 Search Operators and Variation Operators ........................................................................ 94
   4.1.1 Definition of search operators ............................................................................................ 94
   4.1.2 Definition of variation operators ........................................................................................ 95
   4.1.3 Examples of Search operators ............................................................................................ 98
   4.1.4 Examples of Variation operators ....................................................................................... 98
4.2 Practise strategies .................................................................................................................. 100
   4.2.1 Informal description of practise strategies ......................................................................... 100
   4.2.2 Informal discussion of human practise strategies ............................................................... 102
4.3 The variation learning model and its primary data structures ............................................. 102
   4.3.1 Introduction ...................................................................................................................... 102
   4.3.2 Event histories .................................................................................................................. 103
      Events associated with learning progress ................................................................................ 103
      Event histories organize learning events chronologically ...................................................... 104
      Credit assignment for variations within event histories ......................................................... 105
      The state between variation events ....................................................................................... 106
      Branched organization of events in event histories ................................................................. 109
   4.3.3 Section Summary ............................................................................................................ 112
4.4 Variation adaptive loop and its relation to the basic adaptive loop ..................................... 112
   4.4.1 Overview of the variation level ......................................................................................... 112
   4.4.2 Variation Adaptive Loop .................................................................................................. 113
      Introduction .............................................................................................................................. 113
      Brief overview of the variation adaptive loop .......................................................................... 113
      Goal process of the variation adaptive loop ............................................................................ 117
      Affect process of the variation adaptive loop ......................................................................... 119
      Insert Figure 4.8 here .............................................................................................................. 120
      Variation Level Database: Event Histories and Generalized Variation Schemas ................. 121
      Generalized variation schema data structures ....................................................................... 123
      Maintaining a relation, G, between gvschemas and event histories ..................................... 126
      Searching the database and making use of generalization .................................................... 133
      Sorting gvschemas for best one to grow an event history along ........................................... 133
Using temporal difference methods for hypothesizing better gvschemas .................................................. 134
4.4.3 Section summary .......................................................................................................................... 135
4.5 Summary, observations, and hypotheses ....................................................................................... 136
4.5.1 Summary and parallels between the levels .................................................................................. 136
4.5.2 Why variations do not make the "curse of dimensionality" worse .............................................. 136

Chapter 5 Results of the simulation experiments ................................................................................. 138

5.0 Introduction ..................................................................................................................................... 138
5.1 Experiment 1: The basic reaching task ......................................................................................... 139
  5.1.1 Basic results ............................................................................................................................. 139
  5.1.2 Comparison of equilibrium position control vs. direct torque control ............................... 144
  5.1.3 Section summary ..................................................................................................................... 148
5.2 Experiment 2: Learning to reach while the arm is growing ....................................................... 148
  5.2.1 Discretization of the growth process ....................................................................................... 148
  5.2.2 Interactions between growth and learning .............................................................................. 152
  5.2.3 Section summary ..................................................................................................................... 159
Insert Figure 5.9 here. ............................................................................................................................ 160
5.3 Experiment 3: Comparison of minimum-time and minimum-jerk constraints ................................ 161
5.4 Experiment 4: Speed-curvature relationship ................................................................................. 162
5.5 Experiment 5: The basic visual grasp task .................................................................................... 168
  5.5.1 Learning to make saccades to stationary targets .................................................................... 168
    5.5.1.1 Brief description of hardware and software simulation .................................................. 168
    5.5.1.2 Learning modules for the stationary target task ............................................................... 169
    5.5.1.3 Results of experiments with the stationary target task .................................................... 170
  5.5.2 Learning to make saccades to moving targets ........................................................................ 173
    5.5.2.1 Brief description of hardware and software simulation .................................................. 173
    5.5.2.2 Learning modules for the moving target task ................................................................. 173
    5.5.2.3 Results of experiments with the moving target task ....................................................... 175
  5.5.3 Explorations into reasons for sequences of short saccades in infants .................................. 181
  5.5.4 Section summary ..................................................................................................................... 182
5.6 Experiment 6: Learning practise strategies for visual and tactile tasks ....................................... 183
5.7 Temporal difference results ............................................................................................................ 184
5.8 Summary ....................................................................................................................................... 185

Chapter 6 Comparisons with related computational work ................................................................. 187

6.0 Introduction ..................................................................................................................................... 187
6.1 Other computational approaches to reaching and grasping ....................................................... 187
   A Random Sampling Approach to Unsupervised Learning ........................................................... 187
       Overview ..................................................................................................................................... 187
       Learning algorithm ...................................................................................................................... 188
       Conclusion ................................................................................................................................... 189
   An Arm Trajectory Learning Study ................................................................................................. 189


Introduction

The purpose of this work is to come to a better understanding of developmental processes, specifically those in infancy. We have chosen an unusual approach to learning about development. We chose to begin at the beginning and to study some simple visuomotor coordination tasks that are among the first accomplishments of developing infants, namely visual grasp and the reaching phase of tactile grasp. We set out to find or devise enough "formal" developmental theory, and based on the theory to construct a computer simulation of the developmental processes needed to learn the tasks. Computer simulations can be uniquely helpful for showing you where a theory is not sufficiently thought through, or is unworkable for other reasons. This choice of methodology proved fruitful.

To find a starting point for the theory, we were guided by a predisposition toward a constructivist epistemology, and by prior years of following the developmental literature on a number of topics. We had found Piaget's theory of development (Piaget, 1953; Piaget, 1962; Piaget, 1971; Piaget, 1972a,b) to be plausible and particularly comprehensive, if not the most clearly expressed. As we will discuss later in this chapter, several recent attempts to clarify and formalize parts of Piaget's theory have been made, and we were able to build on them in constructing our theoretical starting point. Nevertheless, the theory was still far from what was needed to produce a workable simulation. Many passes were required before theory, algorithms, and simulations produced results.

The thesis is organized into 6 chapters. The first chapter discusses metatheoretical issues that directly influenced our theoretical and computational approach. It also maps out what parts of the theory are from Piaget, or others, and what parts of the theory we supplied. Chapter 2 reviews recent developmental experiments regarding infant reaching movements and infant eye movements. These experiments influenced our research directions in several ways. Not surprisingly, some experimental findings had an immediate influence on the theory. However, what more often happened was that new questions were raised by the experimenters in their research reports, some of which we felt we could address with specific simulation experiments. Such open questions were noted within chapter 2, and the corresponding simulation results are presented in chapter 5. Chapters 3 and 4 present the theory, mathematics and algorithms for the visual grasp and reaching tasks. Finally, chapter 6 contains a discussion and comparison of recent computational studies that are related to the work reported here. Chapter 6 also contains some summary and closing remarks.

---

1 However, Piaget has many interpreters who have often done him the service of improving the theory in the process of clarifying his position on a given issue. Among these interpreters are D. Elkind, J. Flavell, W. Kessen, K. Nelson, I. Fast, M. Bickhard, H. Gruber and E. von Glaserfeld, to name but a few.
Chapter 1  Theory and metatheory for a developmental model of sensorimotor coordination

1.0 Introduction

In this chapter we are going to discuss alternative metatheoretical assumptions for psychological theories. By metatheory we mean those assumptions that form the basis upon which theories in a particular discipline are constructed. In other words, metatheoretical assumptions lie at the foundations of a discipline, constraining the types of theories that can be constructed. In undertaking a brief exploration into psychology's foundations, we are taking a hint from mathematics and physics, both of which have confronted their foundations earlier in this century.¹

The subject matter for our discussion emerges from a comparison of Piaget's constructivist meta-theory with that of empiricist psychology. One reason that this particular comparison is fruitful, is that the two meta-theories disagree on so many points. As we will shortly see, the reason for this disagreement is that empiricism and (Piaget's) constructivism express opposite epistemologies. The empirist view is that the world structures knowledge, not the mind, which only copies it. On the other hand, the constructivist view is that the knower constructs knowledge through his interactions with the world. Furthermore, what is constructed in the knower's head is largely determined by the knower's cognitive apparatus. This knowledge also reflects constraints from the environment's side of the interaction. However, the knower cannot ultimately know the world in an absolute sense, independent of his own acting upon it.

Perhaps, the most important reason for choosing Piaget in the comparison is that Piaget's approach to epistemology has a richness, a coherence and a subtlety that are absent from either empiricism or nativism. While we may not always agree with Piaget's answer to a given question, what has always been impressive is his openness and willingness to continue asking questions, despite the complexity of the developmental or epistemological problem.²

¹ Empiricist psychology has been particularly reluctant to explore its own foundations and this, we feel, has retarded progress in several subdisciplines of psychology for much of this century. Thus, we cannot overstate the importance of studying the foundations of psychology.

² Since Piaget's work is considered somewhat controversial in some circles, it is important that we acknowledge the controversy and provide a brief reply. First, we are using specific parts of Piaget's theory to motivate the learning architecture that we constructed. The merits of this learning architecture can and should be judged by its ability to perform its tasks, not on Piaget's controversial status. The second point we would like to make is with regard to criticism that Piaget's experimental methodology and sometimes his conclusions from the experiments are questionable. Piaget was the first to admit that his experimental methodology was more exploratory than careful. He fully expected and encouraged others to perform the more laborious task of confirming or challenging his experimental results. However, criticisms of his conclusions are usually along the lines of claims that infants possess certain skills much earlier than he estimated. Object permanence is the most controversial point, where Piaget would claim that a fully developed object concept appears sometime between 1 and 1.5 years. Some investigators claim that when infants anticipate the size or trajectory of an object, that this is evidence of object permanence. The reason for the disagreement lies in the understanding of "object" that each investigator has. Those investigators working from an empiricist or neo-empiricist perspective believe that anticipations indicate the presence
What we will do in this chapter, then, is to identify a set of categories of assumptions that we have found from a comparison of various psychological metatheories, and discuss them. In the course of the discussion, we will take a clear position for each category of assumption. Out of our choice of assumptions, a framework that constitutes the developmental theory of the thesis will emerge.

Incidently, we are not taking Piaget's set of metatheoretical assumptions as the last word. There are a few areas where we disagree with some of his assumptions. In these areas we will offer alternatives. One such area of change in meta-theory is in regard to affect and its role in cognition.

1.1 The nature of knowledge

The first area we will address concerns the nature of knowledge. The dominant intuition about the nature of knowledge, that has been maintained in empiricist psychology and is still with us in many guises, is that knowledge is in some sense a copy of reality. In other words, there exists a structural isomorphism between a chair and the image or thought of a chair that is inside the cognizing organism's head.

To come to this view, the British empiricist philosophers Locke (1695/1975) and Hume (1947 edition) drew on Newton's ideas for a mechanical view of the universe. They took external reality as given and asked how the mind could acquire knowledge of that reality. The processes they advocated for acquiring knowledge were mechanical and somewhat reductionistic, but were in keeping with the physics and chemistry of the time. What characterized empiricism was how they defined epistemological problems. For them, the reality was given as well as the assumption of its requiring acquisition/internalization. Consequently, the remaining problem for epistemology was to determine how knowledge was acquired. The nature of knowledge was taken as given, and therefore was not an epistemological problem.

Consistent with these observations, empiricism is usually associated with the "realist" view that there is an objective reality, independent of a cognizing organism. In the learning process, the real object is better and better approximated by the organism's internal copy or representation of the object. This process presupposes that there is a way to compare the copy with the real world. A question, first asked by Socrates, and as von Glaserfeld (1981) points out "has remained unanswered to this day" is, if there is a structured world and if there is a knower whose task is to get to know the world, then how can we ever know that the representation inside the head of the knower is really like the preexisting world? We will explore the implications of this question shortly.

The dominant artificial intelligence (AI) notion of representing knowledge in sentences or propositions in predicate calculus (or some other logic) is closely related to the structural isomorphism view of knowledge. Such linguistic encodings are still of an object concept because to them an object concept is a picture or some kind of structural isomorphism. Piaget's notion of object concept is much more sophisticated than a simple icon. Instead, it is a collection of interactive skills that enable the child to be able to imagine the object's existence when it is completely out of sight (for a period of time) and to be able to formulate a plan to retrieve it that can be quite sophisticated. As we will show in chapter 5, our visual grasp simulations learn an ability to anticipate the location of a target based on its speed and heading, however we would not go so far as to say that this indicates that our system has developed an object concept. It hasn't.
attempts to represent structures and their relationships in the world as some kind of isomorphic copy. As Bickhard (1980a) has pointed out, linguistic encoding only adds another step in the structural mapping from object to representation. This approach can be traced to early Wittgenstein (1933) who theorized that a sentence is a logical picture of the structure of some facts in the world. In order to discover whether this picture is true or false, it must be compared with reality.\(^3\)

There are many arguments against this isomorphic copy notion of knowledge, beginning with Kant (see Kant, 1781/1965 edition). Kant recognized that knowledge was not simply transferred into the brain, but that humans actively construct concepts as they organize their understandings of the world. The notion that mental activity is necessary for knowledge to develop (except for some \textit{a priori} concepts) suggested that the long-held distinction between reason and known reality was not so clear.

Bickhard (1980a) has provided a logical argument refuting the copy notion of knowledge, which can be regarded as a more sophisticated version of Kant's argument. We will present a sketch of Bickhard's argument here. See (von Glaserfeld, 1981) for a very different and imaginative argument that reaches the same conclusion as Kant, Bickhard, and Piaget (1972a).

Bickhard uses a two part argument. He first argues that structural isomorphisms cannot be successful representations, and then he argues that structural representations cannot capture the essence of knowledge.

For the first argument, a structural representation consists of some set of basis elements with some principles of construction. The principles of construction define the isomorphism. In different AI systems, the basis elements may be points, features or logical propositions. Many types of principles of construction have been attempted. Juxtaposition of features, membership in a relation, or logical connectives are some of the more popular constructors. Bickhard points out that the enterprise of finding an adequate set of basis elements and constructors for a domain of knowledge usually founders on one of two difficulties: (1) finding a sufficient set of basis elements and constructors to produce a desired set of representations, or (2) accounting for the origin of the basis elements and constructors. Bickhard's second point is more commonly encountered in the guise of the related problem of providing a mechanism for learning to extend the set of basis elements and constructors so a particular system can represent more knowledge. As the recent history of knowledge representation in AI will attest, each effort to represent a domain produces an \textit{ad hoc} proliferation of basis elements and corresponding construction principles. This may not always be apparent from reading a journal article on a representation project, but first hand experience with such a project always reveals these difficulties. For example, imagine, to capture the general category of "chair." One can imagine a set of features that handle a paradigmatic chair, but what about an unusual chair, such as a large beanbag chair, or an artistically sculptured chair? Humans easily recognize these as chairs, but they are highly problematic for structural representations. These difficulties are compounded with more abstract concepts such as "virtue" or "meditation." Difficult concepts, such as these, can rarely be represented in terms of the

\(^3\) This comparison of reality with the contents of the mind that is regarded as problematic beyond repair by constructivists. Empiricism implicitly assumes that a mechanism for this comparison exists and does not pursue the matter further.
existing basis elements (propositions in this case), and are usually handled by an ad hoc extension of the basis elements with the literals for the very concepts themselves. Once these literals are introduced, constructors that capture possible relations between these literals and others in the basis must also be introduced. The results always have formidable combinatorics and very little explanatory value. Given the difficulties of extending such representations with a knowledgeable human doing the encoding, creating a machine learning system to replace the human encoder cannot be expected to succeed. Arguments along these lines lead Bickhard to conclude that structural isomorphisms cannot be adequate representations.

The second question, of whether structural representations can be the essence of knowledge, Bickhard also answers in the negative. Here again, there are two difficulties. The first difficulty, of course, is that non-structural forms of knowledge can not be represented in a structural isomorphism (e.g. values, abstract concepts). The second difficulty echoes Kant. If a knowing system is to do something with a structural representation, such as generate an action, then the system needs an interpreter of the structure. In Kant's terms, a homunculus is needed to interpret the pictures. However, if the representation is structurally isomorphic to the reality that is external to the organism, then this representation is superfluous, since the interpreter may as well act on the original, rather than the copy. Therefore, the structural representation is not a logically necessary form of knowledge.\footnote{You might counter that an exception is for planning. There it might be very useful to duplicate the world and simulate it. There are many responses to such an argument. The first is that even if planning could be accomplished this way, it still hasn't been demonstrated that this is the way humans do it. The second argument is that there are still very unfavorable combinatorics in the process of making a state machine for the world. Third, a perfectly viable notion of planning exists for the interactive framework that we are about to present. Furthermore, our interactive framework supports a form of anticipation and planning to develop out of basic knowing. Thus, planning is not only a supported process in our theory, but mechanisms by which planning can develop out of knowing are supported as well.}

A variation of the structural isomorphism approach to representation has an incarnation in the neural network literature. This variation tries to build neural network systems that can infer the state machine of some portion of world as a representation. We recommend against this approach because the combinatorics are unfavorable, and because at some level of the system an interpreter is going to be necessary. This interpreter, we believe, must be the subject of psychological study. The enterprise of duplicating the structure and/or state machine of the physical world in simulation must be seen as a postponement of understanding the internal cognitive processes that should be our task as psychologists.

The empiricist program of research was, and still is, to avoid theorizing about what is inside the head, but to study what is outside, instead. This program had the desired effect on psychology. It postponed the study of cognitive processes. The current incarnations of the structural isomorphism approach to representation are holding on to the very same program of research, but in a slightly more sophisticated (and deceptive) manner. The effect on psychology will be the same. Creating simulations of the world as representations will postpone the understanding of the internal "interpreter" (the real cognitive apparatus) of such representations.
We spent some time with this issue about whether copying the world makes any sense as a representation, because it is still the most popular approach taken to representation and we feel that it has not been given adequate scrutiny to reveal its combinatoric difficulties and logical flaws.\(^5\)

To return to our discussion of the nature of knowledge we will interpret and summarize Piaget's position. First of all, Piaget (1972a) identified himself as a constructivist, though others have labelled him (Drescher, 1991) (von Glaserfeld, 1982) a radical constructivist.\(^6\) His work has been about how children construct concepts of object, number, various conservation laws, etc. in the process of actively exploring (interacting with) an environment. In fact, Piaget's use of the term knowledge was radically different from anything in mainstream psychology. He was quite clear that knowledge must not be thought of as a picture or copy of reality. As von Glaserfeld (1982) points out, this was not an assertion that knowledge is an incomplete or distorted picture of reality, but that the nature of knowledge cannot have any iconic correspondence with ontological reality.

Piaget's evidence was that in numerous ways, the child constructs a different reality from the adult. Children's notions of time, space, causality, etc. are not simply degenerate cases of adult isomorphisms. Instead, they have their own structure and go through qualitative changes in structure during development. Not only are a child's constructions of reality different from adults, but the child at different ages will construct a different reality.

Rather than go into the evidence in detail, we will turn directly to a description of Piaget's definition of knowledge. Piaget's use of the term knowledge always involves an organization of operations (i.e. actions) on some object. This organization is not a fixed action pattern and it is not a stimulus-response mechanism. Instead, an organization of operations, a scheme, is a control structure for interacting with the object. This control structure contains some figurative, i.e. perceptual, aspects as well as operative, i.e. (inter)action, aspects, as any control structure must. However, what distinguishes it (radically) from the empiricist position is that the mechanisms for generating and correcting the control structure are both internal to the organism and guided by the organism's needs. The empiricist position is that the environment structures knowledge, not the mind. Piaget's position is that the mind structures knowledge, within constraints imposed by how the environment "reacts" to the organism's interactions.

We can summarize what has been said about the nature of knowledge as follows. The empiricist position puts the knowledge in the world. The task of the organism is to internalize a copy of the structure or state machine of the world. The constructivist view,

---

\(^5\) There is some cause for optimism in that British empiricist philosophy wrestled with the representation problem for about 300 years, American behaviorist psychology pursued the same approach to representation for a little less than a century (so far), and the AI community pursued an analogous approach to representation for about 25 years before giving up on it. It seems that these repeated efforts are having a progressively shorter life expectancy.

\(^6\) The distinction seems to be in the emphasis on how much of the developmental apparatus is provided by genetics. Piaget, being a biologist at heart, used the phrase genetic epistemology to describe his theory to remind us of the biological origins of developmental mechanisms.
as represented by Piaget (1971), von Glaserfeld (1982), and Bickhard (1982), can be summarized by saying that knowledge is the ability to achieve goals. Knowledge of an object, for example, is the ability to interact with the object in a way that achieves a goal of the knower. These two definitions of knowledge have radically different implications for the developmental mechanisms that account for the process of knowing.

1.2 The process of knowing

For Piaget (1971), the process of knowing (intelligence) is an active, interactive, adaptive process. Von Glaserfeld (1982) reminds us that Piaget's use of the term adaptation does not mean that knowledge is made to correspond more and more closely to the external world. Instead, the process of knowing is tied to action. Adaptation means that the organism's (inter)actions increasingly bring it closer to success with respect to some goal. This notion of adaptation is similar to that used in cybernetics.

Thus far, we have equated cognitive structures, in Piaget's use of the term, with control structures for interaction, which he called schemas. According to Piaget, every invocation of a schema has the possibility of being self-regulatory (Ginsburg and Oppen, 1969). In other words, there will be an element of selection of the most appropriate control structure for the goal and specific environmental conditions. This he termed, assimilation. In addition, there may be changes made to the control structure, to increase its utility in relation to a goal. This he termed, accommodation. Assimilation and accommodation are Piaget's basic processes of adaptation.

Furthermore, there are aspects of feedback and repetition in self-regulatory processes. Suppose that the goal of reaching for an object is active, and that a particular instance of a schema, a particular control structure, is selected (assimilation) and executed. If the goal is not achieved, repetitions of the action may occur. Each repetition may involve a modification of the control structure (accommodation), until success is achieved.

We have tried to capture this model of adaptive processes in what we will refer to as a basic adaptive loop, described in chapter 3. In Piaget's terminology, our basic adaptive loop provides the mechanisms for constructing sensori-motor schemas. In our formulation, we have addressed a number of questions within this view of development that have not previously been spelled out. However, these specific questions are all related to the central question for this epistemology which is, what are the developmental mechanisms that construct successful control structures? Sub-questions include: how are control structures constructed, what are possible primitive components of such control structures, what are the allowed modifications to control structures, how are modifications selected, how is success or failure of an interaction determined, how can learning part of a task help in learning the rest of a task?

From a constructivist viewpoint, such as Piaget's and ours, the driving force for an organism's (inter)actions are the organism's goals. This presents a very different understanding of motivation. As Bickhard (1979) noted, the Piagetian view unifies cognition, motivation and learning as aspects of a single underlying process, the process of adaptation. The question of motivation, for example, is the question of which goals are currently active (e.g. hunger, thirst, ...), and how are multiple goals coordinated or prioritized. The process of knowing is the process of being engaged in a successful interaction with something. Furthermore, the developmental, self-regulatory processes,
sketched above, are available during every interaction. In other words, learning and acting are not separated either as separate processes or separated in time, such as with different phases of a process. Every use of a schema involves some change to the schema, some learning, because the basic adaptive processes are always running.

To emphasize a critical point, the constructivist approach to epistemology defines a very different solution to the problem of representation, the problem of what gets constructed inside the head. What is constructed are control structures for successfully interacting with objects, people, etc. (Bickhard, 1982; von Glaserfeld, 1982). That is, knowledge is an interactively competent control structure. There is no more direct sense of knowing than this. Another way to put it is that knowing is instrumental. In this framework, a pattern of interaction with a certain set of invariant properties may constitute a permanent object. Whereas, certain patterns among all objects and their invariances will constitute space and time (Bickhard, 1979).

In contrast with constructivism, for behaving systems based on an empiricist epistemology (e.g. behaviorism) the driving force behind an organism's activity is the environment, via a stimulus. This makes the behaving organism passive and reactive in relation to its environment. All the organism can do is respond in the one way it knows, which has been programmed by the environment by a pattern of rewards and punishments. Thus, the force for correcting an action is via an externally administered reward or punishment. Consequently, the environment initiates and guides an organism's behavior. A recent and clear expression of this point of view can be found in Watkins (1989). Thus, a significant portion of the process of knowing as well as the knowledge are supplied by the environment in the empiricist approach.8

It may now be clear why earlier we identified the empiricist and constructivist approaches as nearly opposite epistemologies. The empiricist approach puts knowledge and teaching in the environment. The constructivist approach, on the other hand, emphasizes the organism's construction of knowledge as driven by its developmental and adaptive mechanisms, but constrained by the environment's side of the interactions (which are not simply related to the structure of the environment).

1.3 The role of affect in learning

As we have discussed, the selection of which control structure to execute is in part determined by the goal. For example, the activity relevant to satisfying a hunger goal is different from the activity for satisfying a thirst goal. However, there is another kind of selection that has not been addressed. This selection has to do with determining when a pattern of interaction is succeeding or failing, and how to assign credit for the success or failure to some specific aspect or step of the interaction. In other words, when a control

---

7 It should be problematic, from an empiricist standpoint, that human's in a sensory deprived environment generate mental activity, rather than simply shut down or go to sleep.

8 Considering how the empiricist approach puts so much structure and process in the environment, and so little in the organism, it should be surprising to proponents of this view that they haven't succeeded in building an artificially intelligent machine. During the 1980's this was, in fact, the dominant program of research taken on by the AI community. The current trend in research, which is away from AI and toward "neural networks", reflects a failure of the approach so long advocated by Newell and Simon, Feigenbaum and others.
structure does not result in success, it is important to be able to determine or at least guess where things started to go wrong. If such sites are labelled, such as with an appropriate affect, then future invocations of the control structure can also be patched with fixes at the sites labelled as problematic. It is our hypothesis that in biological organisms there are mechanisms associated with affects that are responsible for this process of determining whether interactions are progressing toward a goal, or, if not, estimating where difficulties lie.

Since empiricism puts this and other problems in the environment, we will not consider empiricism further on this point. On the other hand, references to children's affect in Piaget's work are like UFO sittings. You hear about them every once in a while but you do not take them seriously. Decarie (1978) has taken the trouble to find these references and deduce Piaget's view of the subject. Piaget found cognition and "affectivity" to be inseparable from one another, functionally parallel, and in constant interaction. He concluded that affect is not an epistemological problem, since it could not be distinguished from cognition, and he chose not to study it further. In disagreement with Piaget's dismissal of affect, our discussion continues.

The insight or hypothesis that guides our explorations of the mechanisms of affect is that affects are physiological signals that tell us how we are progressing in relation to our goals. Simply put, hunger and satiety are negative and positive affects, respectively, associated with the goal of maintaining blood glucose levels. A distinction is to be made between the affect signal, that we associate with the experience of affect, and the affect mechanism, which is responsible for determining when to provide an affect signal, how strong the signal should be, and whether it is positive or negative (for those affects that have two poles). A more abstract way of putting the definition of an affect mechanism is that it is an internal mechanism that is associated with a goal, and which interprets the outcome of actions that have been initiated and selected (i.e. motivated) on behalf of the goal. Though an affect mechanism is internal to an organism, it often must sample the outcome of an action through the senses. However, many times relevant outcomes may be determined quite indirectly by pain mechanisms, or metabolic changes, such as due to exertion or stress. Thus, a lot of internal mechanism may be lurking behind the phrase, "interpreting the outcome." We have some speculations about the affect mechanisms that are relevant to the sensori-motor tasks for the experimental parts of this thesis. A systematic approach to the topic is needed.

In chapter 3, we will discuss hypothetical learning mechanisms associated with the affects of frustration (failure w.r.t. the goal), elation (success w.r.t. the goal), surprise (both positive and negative), pleasure, and pain. These are, perhaps, generic affects, though their mechanisms sometimes have task-dependent features. In addition, we hypothesize that there are task-specific affects that are associated with the tasks of visual and tactile grasp. These task-specific affects may not be prominent phenomenologically, but we believe they must be present in some form for learning to occur.

For example, frustration is an affect that is associated with an expected length of time that a task should take. We experience it when the goal has not been achieved after the usual amount of time spent. Frustration increases in value with time.⁹

⁹ Phenomenologically, "increasing in value" means that the experience is more intense.
We associate default learning-related mechanisms with all affects. Pain is usually associated with marking the schema as failing at the location where pain occurs. This part of the path through the schema will be avoided on future executions. A strategy associated with frustration is to abort the current goal-related activity, mark the schema as unsuccessful, and possibly apply a credit assignment algorithm to help determine why the schema was unsuccessful, if a credit assignment algorithm is available. Frustration is less specific than pain in that it is not always clear where the error that made the schema fail occurred. However, possibilities include sites of recent changes made to the schema or sites where unexpected events may have occurred (producing the affect of surprise).

It is important to mention that mechanisms associated with affects are themselves subject to learning. Both task-specific strategies and general strategies for dealing with frustration can be much more sophisticated than the simple mechanism suggested here. Such mechanisms show great individual differences, which suggests that strategies associated with affects are indeed subject to learning and development. This is an unexplored but promising direction for further research.

Lastly, Bickhard (1980b) presents an interactive model with a similar approach to ours. In his model, positive or negative affects reflect the progress of the interactive learning machine toward or away from a goal. Bickhard suggests that affects are associated with the system's internal sense of certainty or uncertainty in relation to its progress toward a goal. He proposes that a measure of this certainty/uncertainty could involve monitoring how much exploration (i.e. search) is needed as the learning system tries to find its way to the goal. However, the mechanism that determines which affect to generate and whether to label the affect as positive or negative is not clearly specified in his discussion. In other words, he presents the basic insight that affect is related to progress toward a goal, though he does not pursue this insight in sufficient detail to determine its viability.

### 1.4 Learning to learn

Except for our comments on the role of affect in learning, our discussion so far has emphasized that which is in common between our interactive model and Bickhard's or von Glaserfeld's. In fact, the general approach taken by this thesis has been to build upon these two clarifications of Piaget's theory as a starting point. However, as we tried to build a

---

10 Experience with our system suggests that Bickhard's affect mechanism, as he described it, would not work. The difficulty is that early in the process of learning a schema, uncertainty is very high, both for actions that are approaching the goal as well as actions moving away from the goal. Our guess is that his mechanism would prematurely rule out viable paths as well as non-viable paths. To propose a computational model without implementing it and running simulations is somewhat risky because there is no way of determining where your thinking is not complete compared with the range of things that can happen.

11 Another recent attempt to implement aspects of Piaget's work, Gary Drescher (1991) has taken an approach that, interestingly enough, draws more from Piaget's data and observations of development than on his epistemological theory. In fact, Drescher's epistemology has more in common with AI than with Piaget's epistemology. In keeping with the AI approach, the criterion of success of his system was not the system's interactive competence. Instead, it was the existence of certain relational data structures created
simple interactive learning system, a number of difficulties arose that were not in any of the theories. This should not be surprising because none of these theories has been accompanied by a computer implementation of the underlying process model. Resolving the difficulties required explorations into areas that the theories had not anticipated. One of these areas we have named *practise strategy learning*, and it involves an additional layer of structure beyond the usual layer(s) of structure(s) associated with schemas.

Suppose that the system has a set of senses and a set of motors, with dimensions \( s_i \) and \( m_j \), respectively. Suppose also, that a schema is a control structure that, once entered, maps values \( <s,t> \rightarrow a \), where \( s \) is a vector of sensory values of dimension \( s_i \), \( a \) is a vector of action (motor) values of dimension \( m_j \), and \( t \) is time. Among the first questions that need to be answered are: (1) what should the initial control structure be before any learning has taken place, (2) what should the space of possible control structures look like, (3) what are the allowed transformations taking one control structure to a modified version of itself, (4) what are the principles by which these modifications are to be made, and (5) how can successful modifications be structured, and possibly sequenced in time, and retained? These are all different parts of asking, *what is a schema and how does it grow?*

To answer the first part of the question, "what is a schema?", recall that a schema somehow contains the information needed to implement a function of the form \( <s,t> \rightarrow a \). We found that for both implementation and aesthetic reasons a schema is essentially a tree data structure. Since we often want \( <s,t> \rightarrow a \) to be a continuous function, the links of the tree contain data samples that can be used to approximate a continuous surface via, for example, gaussian RBF's (see chapter 3). We used the term "grow" earlier in the context of schema transformation because a schema's growth does turn out to resemble a biological tree's growth in many ways (which resembles axon and dendrite growth). The parts of a schema that get activated repeatedly tend to become more permanent (metaphorically speaking, they thicken with successful re-use).

To answer the second part of the question "what is a schema and how does it grow", recall that a schema is a control structure that guides the organism's interactions with its environment for a particular goal. The schema gets input from the senses and sends output to actuators when it is being executed. A schema is a control structure for guiding interactions at the boundary between the organism and its environment. A *practise strategy*, on the other hand, is a control structure that guides internal interactions. More specifically, the inputs are schemas and the outputs are transformed schemas. The goal of a particular practise strategy is to promote learning of a particular schema. Consequently, while the schema is attempting to guide external interactions to satisfy the

by the system. In other words, the programmer interpreted the data structures to judge what the system learned, rather than the system demonstrating what it knows via its ability to perform some task. Nevertheless, there is much in Drescher's thesis to recommend it, even though it's epistemology is something of a hybrid of empiricism and constructivism.

---

12 More specifically, this is a first level Piagetian schema. A second level Piagetian schema would be a schema that at least in part consists of a composition of first level schemas.
goal, the practise strategy is monitoring the schema's progress and modifying the schema so that the schema becomes more successful in relation to its goal.

To get an idea of what a practise strategy is, without going into too much detail, suppose we have a schema that is a simple step function of time (ignoring for simplicity the sensory inputs), where the actions are hand positions in 3-space maintained from time, t, for duration, d. Thus, the simplest schema structure would be one of the form

\[(x_1,y_1,z_1,t_1,d_1), (x_2,y_2,z_2,t_2,d_2), ..., (x_n,y_n,z_n,t_n,d_n), \text{ where } t_1 < t_2 < ... < t_n.\]

We call the individual schema transformations within a practise strategy variation operations, inspired by the notion of variations in music. One such variation operation for the above sequence might be to tack on another movement at the end of the sequence, \[(x_{n+1},y_{n+1},z_{n+1},t_{n+1},d_{n+1}).\] Another variation operation might be to split one element into two elements of half the duration. If this were done for the first element, it could yield the sequence,

\[(x_1,y_1,z_1,t_1,d_1/2), (x_1,y_1,z_1,t_1+d_1/2,d_1/2), (x_2,y_2,z_2,t_2,d_2), ..., (x_n,y_n,z_n,t_n,d_n).\]

A more common operation is to modulate one or more of the actions in the sequence by a little bit. For example, changing the last position in the sequence by a random vector of the form, \((e_1,f_1,g_1,0,0)\), is called a mutation operation and would yield,

\[(x_1,y_1,z_1,t_1,d_1), (x_2,y_2,z_2,t_2,d_2), ..., (x_n+e_1,y_n+f_1,z_n+g_1,t_n,d_n).\]

Notice that the 3 operations listed are sufficient to generate finite sequences with an arbitrarily fine temporal mesh, but starting from a coarse temporal mesh.

What came as a very nice surprise is that the same basic adaptive process that works at the schema level, also works at the practise strategy level. In other words, the algorithmic architecture for learning a schema is isomorphic to the algorithmic architecture for learning a practise strategy. Furthermore, a practise strategy, as a control structure, is essentially a schema with different inputs and outputs. A more detailed description of the practise strategy learning algorithms can be found in chapter 4.

A comment should be made so as not to confuse Piaget's notion of levels in schema structures (which give rise to stages of development) with the practise strategy level(s). If schemas at the first level can be sequenced to solve a larger, composite task, whose subgoals are the goals of the first level schemas, then the system potentially has the ability to form plans and execute them. Such a system has two knowing levels. In other words, control structures for interacting with the environment would then have two levels of structure. Practise strategies can be associated with each of these levels, though neither practise strategy is actually part of a first level schema or a second level schema (i.e. a plan).

Practise strategies are additional and separate levels of control structure in the system. The practise strategy levels are parallel to and in 1-1 correspondence with the schema levels. If a schema is thought of as having many branches, and if one branch of the
tree is learned, then the practise strategy for that branch can be used to hasten the growth of new branches of the schema. In other words, the practise strategy contains the history of successful transformations that were used when the first branch of the schema was learned. It can then be applied, without so much trial-and-error, to new branches (see chapter 4 for more details).

As far as we know, the notion of practise strategy does not occur in Piaget's theory, though the notion is not new to psychology. However, no prior proposals have been made about what practise strategies are and how they can be learned. Consequently, our collection of architectural components, learning algorithms, implementation, and simulation results constitute a novel approach to the topic.

1.5 Summary

In this chapter, we have outlined some of the theoretical and meta-theoretical issues that are crucial for understanding the motivation for this thesis. In some cases, meta-theoretical assumptions were discussed because they form the foundation on which the theory of the thesis rests. Assumptions concerning the nature of knowledge and the process of knowing fall into this category. In other cases, theoretical constructs were invented or discovered as we attempted to take an initially crude theory and construct a computer implementation from it. Our analysis of the role of affect in learning, and the mechanisms of practise strategy learning fall into this category.

The starting point for our theory and meta-theory has been a careful interpretation of Piaget's approach to epistemology and his theory of development. We chose Piaget's approach as a model both because his ontology is constructivist, and because his psychological theory takes an interactive approach to the mechanisms of development. Our interpretation of Piaget is consistent with two other mathematically oriented researchers: Mark Bickhard and Ernst von Glaserfeld.

We have discussed arguments that suggest that either a copy notion of representation or the process of inferring a state machine of the world as intractable approaches to representation, both because they still do not address the interactive capabilities of the organism, but also because these forms of representation are combinatorially very unfavorable. On the other hand, we recommend an interactive approach to the problem of representation, and a consistent focus on what is inside the head, rather than a focus on what is outside the head. This thesis will be one example of what can be done with such an approach.

We have introduced a new view of affect and its relation to learning and adaptation. Our definition of an affect mechanism is that it is an internal mechanism that is associated with a goal, and which interprets the outcome of actions that have been initiated and selected on behalf of the goal. If this view is correct, then affect must be studied in relation to learning, and equally importantly, learning cannot be studied without also studying affect.

We have also introduced a new category of cognitive structures, not previously considered in developmental theory, called practise strategies. They are a kind of control structure (i.e. a kind of schema), that systematically modifies sensori-motor schemas so that the sensori-motor schemas improve in their ability to satisfy their goal. Chapter 4 contains a description of a set of mechanisms for learning practise strategies.
Lastly, there is a beautiful analogy made by von Glaserfeld (1981) that summarizes the status of constructivist epistemology.

"Piaget has observed that not only the child in his or her ontogenetic development moves from egocentricity to states of increasing decenentration, but so does our species. Looking at our intellectual history and the progression of cognitive constructs and explanatory models, Piaget singled out Copernicus who successfully abolished the egocentric notion that the little planet on which we live must be the center of the universe. We know that was a difficult step to take and that resistance against it lasted longer than a century. It seems that now there is yet another, even more difficult step in that direction we shall have to make, namely, to give up the notion that the representations we construct from our experience should in any sense reflect a world as it might be without us."

To compare our progress with our counterparts of Copernicus' era, Piaget's early formulation of a constructivist viewpoint appeared in print during the 1920's. Whereas, it took about a century for Copernicus' ideas to take hold. While there is now a resurgence of interest in constructivism and Piaget's interactive framework, constructivism still does not have the strong following that empiricism, in its various guises, continues to have.
Chapter 2 The Development of Visual Grasp and Tactile Grasp in Infants

2.0 Introductory remarks on the developmental findings for vision and motor skills

The developmental efforts reviewed in this chapter will be experimental in nature. Our attention will be restricted to experimental findings in the two categories related to our computational experiments. These categories are reaching movements and eye movements. In the area of reaching movements, we will discuss both infant (roughly up to 1 year) and adult findings. Our concentration on infancy is consistent with the focus of this thesis on early developmental mechanisms. The adult studies are included for comparison.

Probably because of the availability of technology and experimental methods adequate to the task, there has been an increase in the number of quantitative studies of infants within the last few years. Several aspects of these studies, particularly the reaching studies, may be surprising. The first is that the recent studies often disagree sharply with the findings of studies done during or before the early 1980's. Discrepancies occur even with qualitative descriptions of basic activities, and developmental sequences. Second, labs doing what appear to be the same experiments with adequate methodology and technology obtain remarkably different results. Third, even the less controversial results are sometimes quite counterintuitive and contradict widespread assumptions about how development proceeds.

In the process of presenting the findings and controversies, we will note where we feel we can suggest an explanation and a computational experiment that can shed light on the matter. In chapter 5, when we present the results of our simulations, we will refer back to the developmental issue that provoked the experiment.

2.1 The Developmental of Reaching and Grasping in Infants
2.1.0 Chronology of developmental events for arm movements

In this section we will summarize some of what is known about the development of reaching movements in infants. Our discussion of the physiology of reaching movements will be somewhat limited, since the physiology has been systematically studied in adult primates rather than in human infants. See (Georgopoulos, 1990) for a review of the physiology in non-human primates.

The literature on the development of reaching movements in infants is relatively sparse for the ages of birth to 4 months (White, and Held, 1966; White, Castle and Held 1964; Hofsten, 1990; Piaget, 1953). On the other hand, there have been many recent studies of reaching movements for the age range from 4+ months to about 1 year (see von Hofsten, 1990 for a review and chronology of developmental events). The reason for this emphasis has probably been the general agreement that observably effective reaching begins in the fourth month (Mathew and Cook, 1990). Among these studies, there is an increasing emphasis on quantitative approaches (e.g. Mathew and Cook, 1990; Fetters and Todd, 1987; von Hofsten, 1991).

According to historically early accounts, the developmental time course of reaching is as follows. Initially, infants do not seem to be aware that their hands and arms are under their control. They seem surprised when their hands accidently travel within
their gaze (Piaget, 1953). As they begin to appreciate that they can control their hands, they produce swiping movements at visual targets, but without visual appreciation of the hand's starting location (White, et al., 1964). Vision is used primarily to site the target, not to monitor the position of the hand before or during the movement.

Initially, these "prereaching" movements appear to be inaccurate, single thrust swipes at the object, preceded by a visual siting of the target, and sometimes followed by a collision of the hand with the target. However, there does not appear to be any visual regard of the hand during the movement. Early movements are stiff and abrupt. With practise these single swipes at the target put the hand closer to the target (von Hofsten, 1982). At some point, it has been assumed that infants begin to alternate looking at the hand in its resulting position after the swipe, with looking at the target to generate the next corrective attempt (White, Castle, Held, 1964). Thus, it was believed that after bringing both the hand and target into view, infants could begin to chain together several corrective reach attempts. This has been regarded as the beginning of visually guided reaching and grasping. However, exactly when visual siting and visual, as opposed to tactile or proprioceptive, interpretation of the result of the reach begins to occur, has not been conclusively demonstrated.

Some recent findings (Clifton, Muir, Ashmead and Clarkson, 1992; Ashmead, McCarty, Lucas, Belvedere, 1992) call into question the assumption that visual guidance (i.e. alternately comparing hand location with target location) is either necessary for learning, or necessary for making early reaching movements, beyond the initial visual siting of the target. Indeed, it has not been clearly demonstrated that infants can interleave visual sightings and visually induced motor corrective commands to guide and improve reaching movements. We will discuss the uses of visual, tactile, and proprioceptive information during the reaching task, as well as the learning issues raised by these questions in the sections that follow.

As mentioned earlier, infants do not make arm movements that are self-evidently target-directed until some time in the fourth month, though some investigators have reported evidence of target relatedness in the spatial distribution of arm movements made by infants as young as 1 week (e.g. von Hofsten, 1982). However, there is not general acceptance that object-directed arm movements occur before the obvious emergence of reaching in the fourth month (Mathew and Cook, 1990).

Beginning around 4.5 months, infants are able to make somewhat reliable reaching movements that contact a target object, though a successful grasp does not reliably follow. Thereafter, the movement becomes more accurate, and more linear, though the time period over which linearity improves is in dispute. Between 4 and 5 months, human

---

1 This is not completely surprising, because infants' earliest reaching movements are entirely proprioceptive. Putting the hand into the mouth is learned by the neonate, usually with eyes closed or looking in another direction (Butterworth, 1986; reviewed in von Hofsten, 1990).

2 As we will discuss below, Goodale, Pelisson, and Prablanc (1986) question whether adults use more than one visual siting of the target for accurate reaching. Our simulation does not actually require more than the initial visual siting to locate the target. Success of the reach is determined by tactile contact with the hand, as appears to be true for infants.
infants begin to show reaching movements that are adapted to the direction, distance, and the motion of the object, although they are probably not able to reach ahead of it if it is moving. The ability to reach toward a visual target is well established by the end of the fifth month (Hofsten and Lindhagen, 1979). By this age, infants can direct their hands accurately enough to contact and reliably grasp objects within their reach, even if the object is moving.

It should be mentioned that it is not clear whether success of a reaching attempt is determined primarily by physical contact with the object, which activates one of several grasping reflexes, or whether success of the reaching attempt can also be determined visually, in the early stages of the learning process. Our simulations, the results of which are discussed in chapter 5, assume only that the movement is terminated by physical contact with the object, and that part of what the infant learns over time is how to use its visual system to determine the relation between its efforts to move its limbs, and where the limbs actually go. That is, she learns to coordinate visual sampling with motor acts and tactile sensations, and she learns how to use vision both to target movements, and later as an estimate of error during the movement at presumed visual sampling moments.

In the following sections, we will review several quantitative studies of infant and adult reaching, as well as studies that explore the learning issues raised in the preceding paragraphs.

2.1.1 Reaching onset and accuracy
Infant studies

Assessing the onset of reaching movements can be difficult because early infant arm movements look like random thrashing. You cannot easily be sure whether the infant is reaching or merely moving the arm about. While von Hofsten (1982; 1990) reports that newborns make arm movements somewhat target-directed in the gross spatial distribution of arm movements (e.g. within 32 degrees of target direction), other researchers have confined their investigations to ages where infants can touch a target that is placed before them.

For example, (Clifton, Muir, Ashmead and Clarkson, 1992) considered reaching movements of infants between 6 and 25 weeks old. On average, infants first contacted the object at 12 weeks. However, the same subjects first grasped the target at 16 weeks. A grasp was unambiguous, since it involved full palm contact. Consistent with this result, in a study by Fetter and Todd (1987), which made several quantitative assessments of infant reaching movements, it was determined that among the subjects who were tested at 5, 7, and 9 months, the 5 month olds would miss the target about 26% of the time, touch it

---

3 Adults are well able to site the target visually, close their eyes and make an accurate and smooth reach. Furthermore, deafferented monkeys can also make reasonably accurate movements after sitting the target visually, indicating that movements can be rapidly played back without the sensory part of the loop.

4 Hofsten (1990), who claims that infants do reach ahead of moving objects, and Mathew and Cook (1990) disagree on this point. Our simulations of reaching or eye-head movements toward moving objects may clarify the issue somewhat, since we will show that what looks like an anticipatory movement can, in fact, be an exploratory alteration of the movement program (schema) which happens to put the hand or head/eye in front of the target. When explorations that work are saved as a practise strategy, the beginnings of anticipatory movements may have developed out of active explorations.
about 65%, and grasp it about 1%. A miss was a reach attempt that was judged to be
directed toward the target. The 7 month olds showed fewer than 1% misses, 68% touches
and 31% grasps. Finally, the nine month olds showed no misses, 23% touches and 68%
grasps. In another study, Mathew and Cook (1990) found infants at 4.5 months
contacting the target 71% of the time, at 6 months contacting it 81%, and 7.5 months
95%. In addition, they ran adults on the same task, who, as expected, contacted the target
100% of the time.

Hypotheses amenable to simulation (set 1)

Many infant sensori-motor skills are believed to develop "out of" basic reflexes. In
other words, reflex actions provide the initial movement attempts from which trial-and-
error information is obtained. To put this another way, at least some reflexes may be
built-in bootstrap mechanisms for the sensori-motor learning apparatus. Von Hofsten's
observation that even infants' earliest arm movements, though erratic, are roughly target-
directed, is consistent with this view that the reflexes give way to crude, but nevertheless
directed movements which are a manifestation of a learning process that is already active.
We will explore this point of view with the first simulations presented in chapter 5
(experiment 1).

There is another aspect of early development that should be accounted for, if we
are to argue that our simulations are developmentally plausible. This aspect is rapid
growth, especially of the limbs. We have sought to construct a learning algorithm that
progresses, even while the limb in increasing in size and mass, and where stiffness and
viscosity about the joints may change as well. Experiment 2 in chapter 5 demonstrates
that learning can occur during growth of the limb using our learning algorithm. The
experiment also explores the conditions under which learning can proceed.

2.1.2 Organization of movement into units

Infant studies

Among the more interesting quantitative aspects of infant reaching is the
segmentation of movements into what have come to be called movement units. We will
compare 3 recent quantitative studies where an attempt was made to characterize infant
arm movement trajectories, infer control strategies from the characterization, and
demonstrate or refute any developmental changes in the characterization. For each study
we will note: (1) what the task was, (2) what criterion was used to delineate a movement
unit (MU), (3) what the structure of the trajectory was for this task, (4) what
developmental changes were observed, (5) what the authors believed their data was saying
about the arm control function.

The first study is a replication by von Hofsten (1991) of his own earlier studies. In
order to explore the structure of reaching movements, he ran 5 infant subjects
longitudinally from 19 weeks to 31 weeks of age. The task consisted of an object moved
manually along a circular path and stopped within reaching distance of the infant. Infant
reaching movements were monitored with a selspot device. Trials were started when the
infant's arms rested at her or his lap.

The criterion von Hofsten used to determine the boundary between two movement
units was a deceleration of an LED on the hand, followed by a reacceleration. Thus, a
movement unit was characterized by an acceleration of the hand followed by a
deceleration. He found reaching trajectories to be relatively straight within a unit, with
most of the direction change in the trajectory occurring during the speed valley between
units. At all ages within the study there was one relatively large (in time and distance) unit
that dominated the reach, which was called the "transport" unit. Hofsten found that the
transport unit increased in duration from 300 msec at 19 weeks to 600 msec at 31 weeks.
Furthermore, the other movement units in the reach remained at 200 msec each. The
number of movement units for the entire reach declined with age from 3 to 5 per reach at
19 weeks, to 2 at 31 weeks. Concomitant with these changes, the transport unit was more
likely to be the first unit in the reach at 31 weeks than at 19 weeks: at 19 weeks it was the
first unit in half the reaches; at 31 weeks it was first 80% of the time. Neither the duration
of the entire reach nor the peak velocity during the reach changed, on average, within the
time span of the study.

Mathew and Cook (1990) examined movement units in greater detail to see
whether they could find evidence of a control function within their bounds. They used 30
infants in 3 age groups, 4.5, 6 and 7.5 months. In addition, they ran two adults for
comparative data. The task consisted of objects which would appear from behind a curtain
composed of strips of black cloth. Two cameras were used to record the infants' movements:
one for a top view and a second for a side view. The intention was to elicit reaches to various target positions from various starting positions in order to get a
sufficiently heterogeneous collection of movements so that initial aiming and subsequent
correction could be assessed.

Mathew and Cook (1990) separated reaching movements into units in a way
similar to von Hofsten, except that they added an extra layer of structure. They used a
period of continuous motion bounded by local minima in the speed vs. time curve as the
lowest level movement unit. To this they added what they called a movement segment,
which was a period of continuous motion occurring within a movement and bounded by
pauses of the hand (i.e. places where hand speed bottomed to zero). Their findings were
also in line with von Hofsten's. That is, as age increased, the number of segments within
movements would decline from 2.9 at 4.5 months, to 3.1 at 6 months, to 2.0 at 7 months,
and with 1.0 for adults. Similarly, the number of movement units within movements
decreased from 5.6 at 4.5 months, to 5.0 at 6 months, to 3.1 at 7.5 months, with 1.0 for the
adults. Consistent with their segment data was a finding that there was a reduction in the
number of direction reversals within the infant movements from 1.6 at 4.5 months and 6
months, to 0.4 at 7 months, with 0 for the adults.

The third quantitative study of the structure of infant movements that we will
consider is (Fetters, and Todd, 1987). These authors filmed the arm movements (3
cameras at 100 Hz) of 10 subjects at ages 5, 7 and 9 months. The infants were seated at a
specially designed table and chair arrangement that partially surrounded them and inclined
the infant at 70 degrees. From this position the infant would reach for a stationary object
placed in front of him on a table. A reach began with the first detectable movement of the
arm and ended when the hand contacted the object, or when there was a near miss after
the arm was clearly directed toward the object. Other movements were not analyzed.

A movement unit was defined by inflection points in the reach where a local speed
minimum was within 20 ms of a local curvature maximum. Out of 425 curvature peaks
found in 128 reaches, only 12 did not have corresponding speed minima. As their data showed, curvature peaks are much more pronounced and easier to locate than speed valleys. This makes the delineation of the movement unit more accurate and reliable.

The movement units found in this way were clustered around a mean of 200 ms for all age groups. So, unlike von Hofsten (1991), or Mathew and Cook (1990), they found no differences in the duration of movement units either within a reach, or with age. Similarly, they found no developmental pattern in the number of units used in a reach for the different age groups. There were slight trends in their data, in the direction of the other studies, but they were not statistically significant.

Fetters and Todd regarded their results as having confirmed the notion of a movement unit as identified by von Hofsten (1979). However, they found tremendous intra-subject variability in these tasks, with unexpectedly chaotic limb movements, in general. They found it remarkable that a constant duration unit and a speed-curvature relationship were present in the data. Lastly, one point in common with the Mathew and Cook (1990) study was that they found large changes in hand direction at movement unit boundaries, though they did not find a decline in direction changes with age.

Hypotheses amenable to simulation (set 2)

The data obtained by Fetters and Todd vs. that obtained by von Hofsten, and similarly Mathew and Cook, differ markedly in the number of movement units, the duration of movement units, and the developmental pattern of change in the organization of movement units. These differences have been noted in the most recent papers by all 3 groups. Fetters and Todd proposed that the differences are probably due to subtle differences in the tasks, suggesting that the von Hofsten studies used targets that were moving, which would induce the infants to try to move more quickly and knock the target down. We have found similar differences in our simulations (see chapter 5, experiment 3 for discussion and results) corresponding to whether the learning system is oriented toward minimizing time to reach the target, or whether it is oriented toward minimizing jerk (the derivative of acceleration). In the former case, the results mimic von Hofsten's finding, including the developmental trend toward dominance of the transport unit. In the latter case, the results can resemble Fetters and Todd's data provided the joints are given extra stiffness (see also experiment 2 in chapter 5).

Adult studies

Recall that von Hofsten found that the reaching movements of the oldest infants in his studies was characterized, on average, by 1 acceleration and 1 deceleration. This kinematic pattern corresponds roughly with the pattern found for adults. For example, Morasso (1981) reported that the hand trajectory of 2 degree-of-freedom movements (shoulder and elbow) has a remarkably clean single velocity peak (1 acceleration followed by 1 deceleration) for a very wide assortment of trajectories within the workspace. The workspace was limited to the horizontal plane at the shoulder and by the length of the arm. These movements were not terminated with a grasp, so there was no hint of another velocity peak.

When a grasp follows the reaching movement, as in Jeannerod (1981, 1984), there are two distinct phases to adult movements. The first phase represents 70-80% of the
movement, is followed by a discontinuity, where the tangential velocity tends to become constant or to increase. That is, the first phase transports the hand to the vicinity of the target, whereas the second phase opens the hand, orients it and corrects the trajectory toward a successful grasp. The corrective activity of the second phase is not dependent upon visual feedback.

2.1.3 Linearity, duration, and peak velocity of reach

Infant Studies

Hofsten (1991) used, as a measure of straightness, the arc-length of the reach divided by the Euclidean distance between the hand's starting position and the target. By this measure he found reaches to improve in straightness from 2.2 at 19 weeks to 1.3 at 31 weeks. Since he found movement units to be fairly linear, with trajectory changes occurring at the boundaries between units, this straightness improvement coincides with the observed decline in the number of movement units discussed above. His interpretation of these results was that the linear portion of a trajectory was essentially "ballistic" with corrections in direction made between movement units. Neither the duration of the entire reach nor the peak velocity during the reach changed, on average, with development. It should be mentioned, that while the average duration was found to be relatively constant, the duration of individual reaches varied markedly.

Mathew and Cook (1990) used a similar measure of linearity and found it to improve from about 2.6 at 4.5 months, to 2.0 at 6 months, to 1.6 at 7.5 months, to 1.2 for adults. They also found that there was a reduction in the number of outright direction reversals within the infant movements from 1.6 at 4.5 months and 6 months, to 0.4 at 7 months, with 0 for adults. Also consistent with von Hofsten was a finding that the duration of the movement, and the peak velocity did not change significantly in the period studied, although adults showed 1/2 the movement duration with only slightly increased peak velocity.

With regard to linearity, Fetters and Todd (1987) used the same measure of the actual distance travelled divided by the shortest distance as in the previous studies. Unlike the previous studies, they found no improvement in the age range from 5 months to 9 months. All of the infant reaches in their study had an arc-length of about twice the distance between the starting position of the hand and the object. Furthermore, the shape of the paths of all infant age groups were found to be somewhat erratic, especially compared with adult paths. While the duration of each reach was comparable on average to adult reaching movements, the variation was quite high at roughly 10 to 1 (1680 ms down to 140 ms). In line with the previous studies, Fetters and Todd did not find a change in duration or peak velocity with age. In fact, not a single infant demonstrated faster movement times with age. On the contrary, the older infants showed slightly longer movement times. The older infants decelerated and grasped the objects, whereas the younger ones knocked them over, which takes less time.

---

5 They suggest that this may be due to a higher frame rate of 100Hz vs 10 (or 60) in von Hofsten's study. Fetters and Todd also observed how very slight changes in the structure of the experiment could radically change the infants' behavior. Thus, task differences could well account for the surprizingly different findings of each of the labs who have done similar experiments.
Hypotheses amenable to simulation (set 3)

As we suggested in the previous section, we can try to model the von Hofsten and the Mathew and Cook data with a learning system that is oriented toward finding the minimum time to hit the target. On the other hand, the Fetters and Todd data can be modeled with a minimum-jerk constraint. Two simulations, one using a minimum-time constraint and one using a minimum-jerk constraint, are compared in chapter 5 (experiment 3, section 5.3). The results of the two simulations show a pattern similar to the infant data. The minimum time simulations produce reaching movements that tend to achieve linearity, at least between the starting point and the target (not necessarily where the hand ends up), with less practice than the minimum jerk simulations. However, in the long run the minimum-jerk movements tend toward very high linearity (again, see chapter 5 for details and results).

Adult Comparison

A study discussed in the previous section (Morasso, 1981) not only showed adult movements to have a simple velocity profile when velocity is measured at the hand, but it also showed that the trajectory of the hand is quite straight. A related study (Abend, Bizzi, and Morasso, 1982), confirmed and extended these results in an interesting way. Using the same apparatus as in the Morasso (1981) study, Abend, et al. gave subjects different instructions regarding the movement they were to make toward a stationary, illuminated target. When subjects were told only to move their hands to the target (not how to move), they chose relatively straight trajectories, whose speed profile was bell shaped. This corresponds to a one movement unit reach. However, when asked to produce curved reaches toward a stationary target or when required to move around an obstacle, adults often produce a path that appears to be composed of very low curvature (i.e. relatively straight) trajectory portions. These relatively straight portions were bounded by speed valleys, for which there were corresponding peaks in the curvature. By the usual criteria, these portions were equivalent to the movement units of the infancy investigators. Occasionally, adult subjects would make a curved movement that did not have a segmented appearance, but instead had a single peak velocity curve and no clear curvature peak. We will discuss this study further in the next section.

2.1.4 The curvature-speed relation and inferences about the control function

Infant Studies

In section 2.1.2, we presented the findings from several labs concerning the existence of a subdivision within reaching movements known as a movement unit. This unit was demarcated by a local speed valley in the speed vs. time curve of the hand during a movement. In this section we will look at a correlate of the speed valley in human movements, the local curvature peak of the position vs. time curve. We will look at

---

6 The experiment was conducted in a darkened room so that the hand was not visible. As discussed above, the apparatus confined movements to a horizontal plane at shoulder height. The movements recorded were thus constrained to have 2 degrees of freedom, shoulder and elbow.
evidence that these two aspects of movement trajectories are correlated. We will also consider some implications this correlation has for how movements are controlled.

Although the observation that speed valleys are often associated with curvature peaks was made first in the adult literature (Abend, Bizzi, Morasso, 1982), a similar observation was made with regard to infants in Fetters and Todd (1987). They found that in a sample of 425 curvature peaks from 128 reaches, there were nearby points in the reach where a local speed minimum occurred in all but 12 cases. Furthermore, the curvature peak and speed dip were always within 20 ms of each other. Since their movement units had a duration of 200 ms, this represented a very tight correspondence. Fetters and Todd found this correspondence in the reaching movements of all the ages tested (5 to 9 months), with no apparent developmental change.

Having established the existence of a curvature-speed relationship, several authors have attempted to determine what this implies for the underlying control functions for the limb.

Von Hofsten (1979, 1980, 1990, 1991) assumed that the local minima in the speed-vs.-time curves, which mark the boundaries of movement units, corresponded to control points in the movement where the arm is re-aimed. The movement units between these re-aiming points he called "ballistic" movements. By "ballistic" he probably meant "subject to the control of a pre-set motor program", where he is not specifying further the nature of the control of the muscles. He argued that all of the movement correction occurs during the slowdown. It should be mentioned that Hofsten's original arguments were speculations, he did not present any evidence in support of his claims. Similarly presumptive was the related claim that infants can aim the hand ahead of a moving target (though we will present a mechanism in chapter 5 that appears to accomplish the same end, but is not explicitly anticipatory).

Mathew and Cook (1990) took this as the starting point for an exploration of how control is accomplished in infant reaching movements (parts of this study were described in section 2.1.2). They asked whether the speed troughs, which are inflection points in the trajectory, are the only places where error correction occurs, or whether error correction occurs throughout the movement. More specifically, they looked for evidence of initial aiming of the hand toward the target, as well as evidence that some re-targeting occurred during the movement unit, and some re-aiming between movement units, where von Hofsten had placed the control function.

First, they found that the tangent to the movement trajectory at the beginning of the movement was correlated with the direction of the target. This was taken as evidence that the hand was indeed aimed at the target, though the initial errors in aiming were quite high. Furthermore, they found the same correlation when looking at the beginnings of movement segments\(^7\) and movement units, as well as the entire movement. It should be noted, too, that at the start of the movement, the infants' hands were not in view. Surprisingly, there was no statistically significant improvement in aiming with age between the 4.5 months and 7.5 months span of the study.

\(^7\) Recall that segments are movement units where the speed \(>0\) within the segment, but bounded by speed \(=0\) at the beginning and end of the segment.
Second, Mathew and Cook (1990) performed an analysis that essentially separated the local speed minima (= high curvature) portions of the movements from the low curvature portions of the movements. That is, they separated the speed valley (= inflection) regions from the low curvature interiors of the movement units. By comparing tangent vectors with this partitioning of the trajectory, they were able to separate the contributions to error correction from the inflection regions vs. from the contributions to error correction within the straighter interior of the movement unit. They found, of course, that "the local minima in the speed curve were associated with relatively large changes in hand direction, which improved the bearing of the hand relative to the target." In addition, they found that the low curvature interiors of movement units also "tended to adjust the bearing of the hand movement toward the target."

Third, they found a negative correlation between the distance traveled within a movement unit and the initial direction error of the hand for that unit. They interpreted this correlation as an indication that a control process was terminating movement units in the wrong direction in order to re-direct the hand toward the target.

Fourth, they found positive correlations between the distance traveled to the point of maximum speed within a movement unit, and the target distance; and between the total distance traveled during the movement unit, and the target distance.

These descriptive data are consistent with many possible control schemes. In particular, they do not address the issue of how visual feedback, proprioceptive feedback, or in some ways tactile feedback might contribute to the control. Clearly, visual sitting of the target is important, but is vision used to correct the hand when the target and hand are both in view? Is vision used to "linearize" the movement within the movement unit, or is proprioceptive information used primarily for this function? When is proprioceptive information primarily used, or preferred? Similarly, the end of the reaching movement could be determined by any of these sensory processes. Also, descriptive data such as these are not sufficient to determine whether some aspects of the control run "open loop" or in a pre-programmed way, or whether there is an error-detection->compensation->re-test loop at some level of control. Finally, the nature of the learning process could not be addressed, especially as they were not able to find developmental trends in their correlations.

We will begin to examine specific aspects of the control function in the next section (2.1.5).

**Hypothesis amenable to simulation (set 4)**

There are at least two ways we know of producing a speed-curvature relationship using our learning framework. One way would produce the speed-curvature relationship as a side effect of trying to generate a movement toward a target, but through a via point.8

---

8 The via point can be used to find two linear trajectories, one ending near the via point, and the other beginning with the via point. The learning system can use the concatenation of these two trajectories as the starting control sequence for additional learning under the minimum-jerk constraint. After a sufficient number of iterations of practice, the result should show a speed-curvature relationship that progresses from infant-like (speed at inflection near zero) to adult-like (more energy conserving) properties. However, more important than duplicating the kinematics of the relationship, would be if our learning system could learn what transformations of the control sequence produced the speed-curvature
The second way of producing a speed-curvature relationship is to simulate learning under presumptive infant-like conditions. In this case, using equilibrium position control, increasing the stiffness of the arm/controller combination, and making the arm grow together provide sufficient conditions for inflections to occur where the speed minimum occurs exactly at the inflection.

In chapter 5 (experiment 4), we will present a discussion and the results of producing a speed-curvature relationship using the second method. We have not simulated the first method, though our experience with the second method suggests that it would work, especially under similar conditions. This would mean using higher than normal controller gains, using equilibrium position control, and substituting the via-point condition instead of using growth to trigger the formation of an inflection in the trajectory.

Adult Comparison

It may seem, at first, that a local maximum in the curvature of the position-time curve would necessarily be accompanied by a local minimum in the speed. Usually, a massive object must slow down more in order to make a sharper turn. However, human arms, particularly infant arms, are primarily energy dissipative, as opposed to energy conserving systems. Energy can be expended or injected in comparatively large amounts. While you may expect adult movements to be more energy conserving, there is nothing about the more erratic movements of infants that suggests energy efficiency. However, another point suggests that a speed-curvature relationship is not to be expected, a priori. The arm-hand system has many links and many degrees of freedom. Such a system is quite capable of generating movements showing different relationships between curvature and speed.

Nevertheless, this relationship has been established in adult arm movement studies (Abend, Bizzi, Morasso, 1982). Adults making planar, two-joint reaching movements toward a target, when asked to make circular movements or to reach around an obstacle, tended to make movements consisting of almost linear segments separated by points where the arm slowed down and made a highly curved path. In the adult studies the slowdown and the high curvature path occurred close together in time.

It should also be mentioned that in the Abend, et al. (1982) study, the authors noted how problematic the speed-curvature relationship is from the current theoretical framework, which regards the generation of the movement as beginning with an imagined path in visual space, followed by a transformation into joint coordinates and then to muscle activity. Nothing in these transformations, it seems, would necessarily produce the observed relationship. However, we will argue in chapter 5 that our developmental model of the reaching task, which results from a different theoretical framework, does indeed produce the otherwise problematic relationship.

relationship, and use these transformations directly when it needs to produce new movements through via points, but with little or no additional practise.
2.1.5 Relative use of visual and proprioceptive input in reaching task

Infant Studies

The descriptive studies of infant reaching movements in the previous sections do not address crucial questions about the nature of the learning and the control tasks accomplished by the developing infant nervous system. In this section, we will examine the comparative use of vision and proprioception in the development of reaching. More specifically, we will discuss when and under what circumstances infants will use visual information in correcting the position of the hand when it is off target.  

As pointed out in (Clifton, Muir, Ashmead and Clarkson, 1992), it has long been believed that visually-guided reaching emerges when infants are from 3 to 5 months old. Piaget (1953) had conjectured that reaching emerged as a result of a discovered correspondence between visual schemas and schemas for tactile grasp. This statement can be taken to mean different things. Some authors have taken it to mean that a comparison of the hand's position and the target's position, when they are both in view, provides an error that can subsequently be corrected with another movement of the hand. Early, somewhat informal, studies appeared to confirm this view (e.g. White, et al., 1964). On the other hand, Piaget's conjecture can also be taken to mean that a correlation between the experience of tactile contact and the experience of visual contact (e.g. the colliding boundaries of two objects) may help the infant reacher learn that in order to bring about the desired tactile proximity, (s)he must first bring about visual proximity. Mechanisms for such multi-modal correlations have received some computational exploration in Drescher (1991).

These alternative interpretations of Piaget's conjecture regarding reaching are neither exhaustive nor mutually exclusive. However, implicit in either view of the reaching task is initial guidance that is primarily proprioceptive, since the hand must be brought to the vicinity of the target. It seems that there has been general agreement that at the beginning of the movement, the hand is not in view.

These observations suggest that several distinctions should be made before we proceed further. As noted in (Clifton, et al. 1992) a commonly made distinction is whether the reaching movement itself is visually guided or visually elicited. To be visually guided, the hand and the target need to be in view more or less at the same time. To be visually elicited, the movement is triggered at the siting of the target. As they are often used, these distinctions are not entirely satisfactory. For example, visually guided usually refers to the part of the reaching movement just prior to the grasp, where the hand is near enough to the target so that they are both in view. However, the portion of the trajectory before the hand is in view is not identified as being visually guided (hence the term is ambiguous for this part of the movement). As another example, when the term

9 An issue addressed by the set of papers of this section, is whether trajectory corrections are made based on a visual comparison of the location of the hand relative to the target, or whether corrections are made based on visual siting of the target alone, with proprioceptive information being the basis of the corrections to the hand's trajectory.

10 As stated, this conjecture presumes mechanisms not precisely formulated, and certainly not explored. We will discuss a more precisely framed alternative to this conjecture in chapter 3, where we will introduce a revised model of some aspects of sensori-motor development.
visually elicited is used, it usually means that the movement is not visually guided, though it seems that it should be possible for a movement to be both visually guided and visually elicited. Thus, care is needed to be explicit about which part of the movement is being discussed, as well as whether any implicit assumptions are being made about the rest of the trajectory.

There is another, more subtle distinction that is often overlooked. This distinction has to do with whether certain sensory modalities are required to make (or learn) a movement, vs. whether the modalities can be used for the movement. For example, in a study that is frequently cited as providing evidence of visual guidance during reaching, namely (McDonnell, 1975), infants learned to grasp a target while wearing laterally displacing prisms. As pointed out by Clifton, et al. (1992), by the nature of the task, infants may have been induced to use the visual modality, where they otherwise may not have used it.

Another interpretive problem with the prism goggles task is that the infants need not necessarily have used vision to provide the adaptations to the task. As will become clearer from the description of our learning algorithms in chapter 3, infants could learn the prism task with tactile feedback after being surprised when the visually-displaced target was not felt where it was expected. A corrective movement could have been learned after a few trial-and-error attempts and subsequently associated with the goggles. Thus, it is an assumption that infants adapt to the prism goggles with visually guided corrections. Clifton, et al. (1992) raise another problem with the prism goggle study. If the study were to rule out the use of proprioceptive-only information, then the goggles would have had to remove vision and allow proprioception to guide the movement. Instead, the goggles produce a conflict between vision and proprioception which would require a recalibration of the correspondence between the relevant topographic maps. It is not obvious how to interpret the results of the study in view of the overlooked complexity.

One way of avoiding some of the difficulties of such studies is to test infants’ reaching capabilities in the dark (Stack, Muir, Sherriff and Roman, 1989), (Clifton, Rochat, Litovsky and Perris, 1991), (Clifton, Muir, Ashmead, Clarkson, 1992), (Ashmead, McCarty, Lucas, and Belvedere, 1992). One of these studies, (Clifton, et al., 1992) is particularly appropriate for verifying some of the assumptions of this thesis regarding arm movements.

In this study, 7 infants were run longitudinally from about 6 weeks to 25 weeks of age. In each experimental session, infants would be presented with objects to reach for in a lighted room, and in another part of the session, they would be presented with glowing or sounding objects in a completely dark room. The infants’ reaches were scored according to whether they touched the target object, or grasped it. Reaches that did not result in some contact with the object were marked as null trials. A grasp was defined as closing the hand around the target and making contact with the palm.

---

11 If the correction were learned and then associated with the goggles, it would be what we call a practise strategy. The notion of practise strategy is defined in chapter 4 and is the central notion of the chapter, which also includes algorithms by which practise strategies can be learned.

12 Testing infants in the dark allows you to turn off vision and leave proprioception on. Unfortunately, there’s no obvious way to turn proprioception off and leave vision on.
The aim of the study was to determine the onset of touching and grasping for light and dark conditions, as well as when touching or grasping became stable for each condition. Stability was defined as touching or grasping on more than half the trials in the session. The results were as follows. The onset of touching the object without grasping it ranged from 7 weeks for one infant to 16 weeks. The onset of grasping ranged 11 to 19 weeks. The average age of onset for touching in the light condition was 12.3 weeks as compared with 11.9 weeks for the dark condition. The average age of onset for grasping was 16.0 weeks in the light condition and 14.7 weeks in the dark. Though the individual differences in age of onset were vast, each individual infant would begin to touch in both the light and dark conditions at about the same time, regardless of the age of onset. The mean difference between light and dark conditions for the onset of touch was 0.4 weeks. The same was true about the onset of grasping, with a mean difference in the light and dark conditions of 1.3 weeks. A similar pattern was true of achieving stability for touching and grasping. In both light and dark conditions touching, which began at 12 weeks on average, stabilized at 16 weeks, whereas grasping, which began at 16 weeks on average, stabilized at 20 weeks.

Curiously, the largest difference in age of onset for touch or grasp in the light vs. dark conditions was a difference of 4 weeks for 3 infants, with the dark onset preceding light onset in all 3 infants. T-tests did not reveal significant differences in the onset times, however. A statistical power analysis was performed and indicated that if reaching and grasping in the light had preceded reaching in the dark by as little as 2 weeks, the difference would have been detected. This was taken to imply that "reaching in the light developed in parallel with reaching in the dark, suggesting that visual guidance of the hand is not necessary to achieve object contact either at the onset of successful reaching or in the succeeding weeks."

Another recent study examined infant reaching using a "double step" target paradigm (Ashmead, McCarty, Lucas and Belvedere, 1992), which produced results consistent with those of Clifton, et al. (1992). The subjects consisted of seven, 5 month old infants, seven, 9 month old infants, and 5 adults. The task was to reach for illuminated objects, always within reach, and in a darkened room. In a control version of the task, the illuminated object remained stationary as the infant reached for it (no-switch task). In a second version of the task (switch task), the infant's hand was detected with an infrared photoelectric arrangement as it approached the target, at which time the illuminated object was extinguished, and a second identical object was illuminated. The second object was displaced laterally with respect to the first, giving the illusion that a single object had changed position. On half of the trials for each task, the infant's hand was made visible with a luminescent marker (band-aid). The results were that on no-switch trials the 5 month olds showed the same (relatively poor) accuracy of reaching both with and without the luminescent marker, on switch trials the 5 month olds without the luminescent marker showed some tendency to correct their movement toward the displaced target, but this tendency was not significant. On switch trials with the luminescent marker there was evidence that no correction to the new object position was made. Thus, the visible hand marker did not affect whether the 5 month olds made a correction toward the displaced target. The 9 month olds showed a rather different pattern. On switch trials, the 9 month olds partially corrected their movements toward the new target location when they wore
the luminescent hand marker, but did not correct their movement without the hand marker. However, on no-switch trials, the 9 month olds reached equally accurately with and without the luminescent hand marker. Finally, adults showed no differences with or without the hand marker on pilot data, so the experiment was conducted without the luminescent hand marker on either switch or no-switch condition. Not surprisingly, adults fully corrected for the displaced target in the switched condition. In summary, only the 9 month olds showed a distinctly improved correction for the displaced target when their hand was visible as opposed to when it was not visible.\footnote{This implementation of the double step paradigm cannot be directly compared with some of the other studies, especially those concerned with more than just the very end of the reaching trajectory. The reason is that the time of occurrence of the apparent target movement was very late in the movement in the Ashmead, et al. study, and the photoelectric beam to detect the hand was placed just in front of the target.}

Thus, infants do not have to see their own hands, either to learn reaching or to accomplish reaching after it is learned. In fact, these results contradict the hypothesis that infants use visual matching of the hand and target locations to learn reaching.

These results complement an earlier study (Clifton, et al., 1991) where it was shown that infants at 6.5 months can accurately reach to noisy, but dark objects in the dark, and in addition can anticipate the size of the object, identified by its noise. Since these abilities appear to arrive with the onset of reaching, it suggests that if the object can be located, at the initiation of the reaching movement, proprioceptive input is sufficient (and presumably necessary in the first 4+ months) to learn to control the reach, provided that a successful reach can be distinguished from an unsuccessful one.

**Hypothesis amenable to simulation (set 5)**

In chapter 5 (experiment 1), we will present simulation results that demonstrate that visual siting, proprioceptive guidance and tactile termination are sufficient to learn the dynamic control of reaching movements. These movements show adult quality (low jerk and ultimately high linearity). Thus, we will argue that our computational model of these sensori-motor processes is sufficient to explain these recent experimental findings.

**Adult Comparison**

In this section we will consider some studies addressing the question of the relative contributions of visual and proprioceptive input in controlling adult reaching movements. More general considerations about what it is that is controlled will be discussed in section 2.2.

We will begin with an observation about the (Abend, et al., 1982) experiments that both predate and motivate the concerns of this section. The observation is that the subjects in the study were not able to view their hands while making movements. Trials were either run in the dark, or a piece of opaque white paper was placed on a horizontal plexiglass barrier just above the arm's horizontal workspace. This observation is important because it indicates that for accurate and surprisingly linear reaching movements, sight of the hand is not necessary.

A similar finding was presented in (Prablanc, Pelisson, and Goodale, 1986), who looked at the relation between the duration of the visibility of the target and the accuracy of the arm movement. In their experimental setup, vision of the hand was blocked during
the movement. These investigators found that when the target LED did not remain visible long enough to be accurately foveated, then reaching movements tended to fall short of their apparent destination. The most accurate reaches occurred when the target remained visible throughout the movement.

In a companion study Pelisson, Prablanc, Goodale and Jeannerod, (1986) (see also Goodale, Pelisson and Prablance, 1986) moved the target after the reaching movement was begun to see whether a corrected reach would occur. In their experiment subjects were instructed to reach for the target when it moved from its normal resting location in the center of their field of vision out to a peripheral location. Typically, subjects would make two or more saccades in order to re-foveate the target. On randomly selected trials, an on-line computer that was monitoring the subject's eye movements would step the target to a still more peripheral location. This extra step (of about 10%) would occur when the first saccade reached maximum velocity, by which time the arm would have begun movement. The results were that, although subjects would undershoot the target slightly in both the 1-step and 2-step conditions, they did indeed correct the arm movement in the 2-step condition by about the right amount. That is, when the target made an extra jump, after the eyes and hand were already moving, then the hand-arm movement was corrected an amount equal to the extra jump of the target. Thus, visual appreciation of the target movement was required, but vision of the hand in relation to the target was not required for making a visually dependent correction to the arm movement.14

These studies combined suggest several conclusions. One conclusion is that vision of the hand is not necessary for reasonably accurate reaching. Nevertheless, reaching is not completely open loop since there is some form of visually guided error-correction. However, it is not at all clear what the mechanism of this error correction is. Flash and Henis (1991) have suggested an interesting strategy for correcting movement processes after the target is shifted. They hypothesize that the initial trajectory plan could continue to completion. Rather than replacing it, they suggest vectorially adding a second, corrective plan that corresponds to the plan for moving between the first and second target locations. This second plan was constructed as a time-shifted "control-like" trajectory from the first to the second target, with initial and final velocities equal to zero, and with an appropriately scaled, "standard" velocity profile. Simulation results confirmed that this idea is workable. One final point requires mention. None of the above adult studies compare reaching with vision of the hand to reaching without vision of the hand. However, earlier studies indicate that an additional phase of correction occurs when simultaneous vision of the hand and target is available, and that this added phase produces the most accurate reaches obtainable. This does not mean that vision of the limb earlier in the movement contributes to a straighter or more accurate movement. We will return to this issue later with our simulations.

14 Curiously, none of the subjects noticed that on some trials the target had jumped to a new location, while they were reaching. The authors interpreted this as showing that the mechanism that maintains the apparent stability of a target in space is dissociable from the mechanism that mediates the visuomotor output directed at the target.
2.2 Adult reaching movements

There are just of few more points to be addressed from the adult research on arm movements that are needed as background for our simulations. These points have not been treated in the infant literature, at least not treated explicitly.

The first point has been made in a body of research that has explored the nature of movement control in biological organisms, e.g. (Polit and Bizzi, 1979), (Bizzi, Accornero, Chapple, and Hogan, 1984), (Bizzi, Polit, and Morasso, 1976) Bizzi et al. Among the points made by these investigators is that motor control exploits the mechanical properties of muscles and joints, and that a manifestation of this exploitation is that movements are commanded as a sequence of equilibrium positions of the various joints involved in the movement. The positions of joints are specified by the relative activation of opposing muscles. This relative activation establishes the relative tension of the muscles, which through the elastic properties of muscles results in the limb "snapping" to a position that balances the tension. See Bizzi, et al. (1991) for a concise summary of this position as well as some recent, and rather direct, physiological evidence for it.

We are mentioning this point in the chapter on developmental issues because we have found, through simulations, that the equilibrium hypothesis can be extended back into infancy, and when this is done the result is greatly enhanced learning of reaching movements. Thus, our simulations provide further support for the equilibrium hypothesis. We will come back to this point in chapter 5, where we contrast simulation results using equilibrium point sequences as the control learned, vs using a sequence of torques as the control learned. We should also mention that simulations done by Flash (1987) show that trajectories resulting from sequences of equilibrium points can show the same kinematic characteristics as are seen in adult reaching movements.

The second point we wish to address in this section has to do with several implicit assumptions about the biological control process in adults that immediately become problematic when viewed from a developmental perspective. These assumptions are that we "plan" an action path in spatial or extrinsic coordinates, and subsequently transform the path through a composition of coordinate transformations into a sequence of equilibrium position commands in the equivalent joint coordinates. This, incidentally, is the model used by Flash (1987) to generate the equilibrium point sequences for her simulations.

There are two implicit assumptions that we would like to call attention to: the first is that adults plan trajectories in the way described, and the second is that infants somehow have this capacity as well. The assumed notion of planning seems to entail a mental picture, or copy, of the current extrapersonal space as well as an image of the limb's position within this environment. The planning process involves imagining where the limb will go and what path through this 3-d representation it will take. The path, then, is converted to a sequence of equilibrium positions for the limb, and then executed. This seems like a lot more computation than should be necessary. For example, a sequence of

\[\text{References}\]

15 When planning is referred to in the literature, there is an implicit reference to commonly held intuitions about cognitive processes. Our aim here is to try to make the assumptions underlying these intuitions explicit so that they can be questioned and amenable to scientific investigation, rather than remaining apart from it.
equilibrium positions of the limb could be associated with the destination of the limb in visual coordinates and the limb's starting position in joint coordinates. This sequence could be retrieved and executed without all the extraneous processing and without representing the world. Our learning architecture takes this point of view, and extends it by proposing ways in which the appropriate sequence of equilibrium points can be learned by trial-and-error.

With regard to the assumption that infants can do this type of planning in a represented world, there is neither evidence that these capacities are available, nor that they are used.\footnote{Linearity of hand movements through space is often taken as "suggestive evidence" that movements are planned in spatial coordinates. This too we can question with our simulations, if we can show that the straightness can come about without planning. We believe that planning itself is a higher level process, which has its own developmental prerequisite processes. Our simulations and theory begin to distinguish planning from what we call level 1 knowing, and how planning evolves out of level 1 knowing.} However, for infants there is little question that their reaching attempts are extremely nonlinear. Nevertheless, a proponent of the planning in extrinsic coordinates approach could argue that infants use a linear internal representation as a reference, or somehow develop a linear internal representation. The burden of proof is with the proponent of such planning, since this assumption requires that the infant either possess or develop heretofore unexplained mechanisms. Our approach does not presuppose these mechanisms, but implements mechanisms sufficient for the reaching task without either planning or an internal representation of the external world.

More specifically, the difference between our approach and that of Flash (1987), for example, is that our learning architecture does not generate equilibrium point sequences by planning in visual space. Instead, it generates equilibrium point sequences by trial-and-error movements toward a target. However, to do this, we are taking advantage of another aspect of reaching, interestingly enough, due to Flash and Hogan (1985). The trial-and-error movements are assessed relative to certain constraints, including a minimum-jerk constraint. Thus, we can arrive at a similar solution to the reaching problem as Flash's (1987) solution, but without the assumption of the added mechanisms, whatever they may be, involved in explicit planning of a trajectory.

\subsection{2.3 The Development of eye movements in infants}

In this section we will review some findings on the early development of eye movements in infants. We have restricted our attention to saccades, OKN, and smooth pursuit. For brevity, we have omitted vergence movements, optical accommodation, and more complex scanning eye movements.\footnote{We are omitting scanning movement, even though our original intention was to include them in the thesis. However we chose to omit the topic both because it is in its infancy, and because our theory and simulations are not quite ready for more complex visual tasks.}
2.3.1 Saccadic eye movements

Infant studies

For saccades to be accurate, infants must adapt the magnitude and direction of the initial saccade to extraocular muscle strength, eyeball mass, and the damping characteristics of the orbital tissues. This information requires trial-and-error attempts to foveate peripheral targets. Apparently, infants come into the world with some help in this matter, since they are sensitive to the direction of motion just after birth (Nelson and Horowitz, 1987) (Aslin, 1987). Furthermore, Aslin (1987) argues that the existing evidence clearly shows that the direction of their saccades appears to be accurate just after birth. Consequently, the primary task early on is for them to learn to get the magnitude of the saccade reasonably accurate. What has been enigmatic for the research community has been that the amplitude of saccades is surprisingly small. Infants will use a sequence of 5 or 6 small saccades toward a peripheral target. Surprisingly, this pattern exists even though they are capable of making large saccades, as is evidenced by studies of spontaneous eye movements in the dark (reviewed in Aslin, 1987). Furthermore, attempts to explain this phenomenon as due to motivation, or to a low quality of visual input, or to an interaction of the head control mechanism with the eye mechanism when the head is held fixed, or even due to cell migration patterns in the retina, have all failed. Lastly, it has been shown that the sequences of 5 or 6 short saccades made at 2 months will continue, even if the target is extinguished after the first saccade of the sequence (Salapatek, Aslin, Simonson and Pulos, 1980). It is important to note, however, that the sequence of saccades was subtly altered by the disappearance of the target. The subsequent saccades did not show any change in magnitude or direction, as would normally happen when the eye approached the target. Thus, the sequence would usually overshoot the target's location before it disappeared. This pattern of multiple saccades was observed in infants at 5 months, the oldest subjects in the studies cited.

Hypotheses amenable to simulation (set 6)

None of the hypotheses concerning the sequences of low-magnitude saccades in the literature had to do with the process of learning to make accurate saccades to targets. Below are several suggestions that involve the learning process that we have explored with our simulations. The simulation details and results will be presented in chapter 5, experiment 5.

1. Using a sequence of small saccades is physically a more conservative strategy with respect to overshooting the target, which could thrust the eye against the limits of its rotational travel.

2. It could be more efficient learning, if the visuo-motor system already has the correct direction, to creep up on the target. In other words, with multiple saccades, it takes more time for the eye to reach its target, but the eye gets to its target more reliably than it would otherwise. With multiple saccades, the

---

18 Each of these saccades has approximately the same magnitude, except the last. All in the same direction.

19 A multiple-saccade sequence is a sequence that has more than 1 long saccade and 1 small corrective saccade.
only saccade(s) that may need correction is the last saccade in the sequence, and its likely undershoot or overshoot will be within a small radius. If a single saccade were attempted, the radius of probable error would be quite large. Consequently, additional saccades might still be needed to get the eye close to the target.

(3) Once the small distances are learned, then creeping up on the target will result in much better overall accuracy during learning, since the system is making use of those parts of the topographic map that are well learned.

(4) In order to learn to saccade to moving targets, it is much more efficient to begin learning by following the target, and later collapsing the sequence of commands to "catch up", than it is to begin with random search to find the best place to "jump ahead" of the target. One might say that the practise strategy of using multiple short saccades makes more sense for learning to saccade to moving targets. We will discuss this issue in more detail in chapters 3 and 5.

Adult comparison

Saccades are ballistic. Once the neural impulse is sent, the saccade is not usually cancelled or altered until after the eye comes to rest. This is different from most other skeletal muscles of the body, whose antagonist pairs contribute to braking. It is believed that there are very few muscle spindles in the extraocular muscles. Saccades are initiated with a very high frequency burst of firing of the agonist muscles whose duration determines the amplitude of the saccade. Antagonists are relaxed in this period. After the saccade brings the eye to its destination a lower rate of firing of agonists and antagonists hold the eye in its new equilibrium position. Saccade amplitude, in adults, is usually within 5 to 10% of the amplitude required for foveation, so a second, smaller corrective saccade is often required. This second saccade follows the initial saccade by 150 to 200 msec.

If the peripheral target changes during the latency period, then the magnitude of the saccade corresponds to (1) the initial target location, (2) the final target location, or (3) somewhere in between (Becker and Jurgens, 1979).

Other quantitative characteristics of adult saccades include (Kandel and Schwartz, 1985):
(1) saccades take place at speeds of 600-700 degrees per second;
(2) onset latency is about .2 seconds;
(3) the saccade itself takes about .05 seconds to complete;
(4) another saccade cannot be initiated until .2 seconds later;
(5) there are hints of continuous feedback control in saccades;
(6) the direction of a saccade is coded by the specific group of motor neurons activated;
(7) saccade amplitude is coded by the duration of a burst of high frequency spikes;
(8) saccade velocity is not under voluntary control;
(9) visual acuity during a saccade is poor.

---

20 This finding is reminiscent of a similar response made to a displaced target during a reaching movement.
2.3.2 Optokinetic nystagmus

Infant studies

Optokinetic nystagmus (OKN) has been observed in infants as young as several hours old (Aslin, 1987). Stimuli for eliciting OKN in infants typically cover most of the visual field. For example, 12 degree per cycle black and white stripes that are moved at constant velocities from 9 degrees/sec. to 40 degrees/sec. are commonly used. Infants that are 1 to 3 days old will show an increase in slope of slow phase OKN up to 25 degrees/sec. However, OKN is asymmetrical w.r.t. nasalward and temporalward directions, and vertical OKN is clearly present in infants at all ages, while horizontal OKN shows steady improvement over first 1/2 year. For infants (Held and Naegele, 1982) used an asymmetry index which was the ratio of slow phase velocity temporalward compared to nasalward stimulus movement. This index increased linearly from .2 in 1 month olds to 1.0 in 5 month olds (horizontal OKN). Infants at all ages show clear instances of vertical OKN.

Adult comparison

Adults show a phase velocity that increases monotonically between 9 degrees/sec. and 40 degrees/sec (Kandel and Schwartz, 1985).

2.3.3 Pursuit eye movements

Infant studies

Unlike saccadic eye movements and OKN, smooth pursuit eye movements are both rare and difficult to measure before the end of the second month. There has been some confusion in the literature concerning the onset and accuracy of smooth pursuit, partly because early studies used sufficiently large targets to elicit OKN, which was probably misidentified as smooth pursuit. However, the technology for eye tracking had to mature from electrooculography to corneal photography before detailed studies of smooth pursuit could be done. The pivotal study on the onset and early development of smooth pursuit in infants was conducted by Aslin (1981). Using an infrared video-based recording system, Aslin tracked infant eye movements between 5 and 12 weeks postnatally. The stimulus was a black, vertically oriented bar (2 degrees by 8 degrees) moving horizontally on a white background. The target oscillated sinusoidally with a 20 degree amplitude. Test velocities from 10 degrees/sec. to 40 degrees/sec. were used. Even at low target velocities none of his 32 infant subjects showed any evidence of smooth pursuit within the first 5-6 postnatal weeks. All of their eye movements in this age range were saccadic. During the 6th to 8th postnatal week, very brief segments of smooth pursuit emerged among the saccades, though the accuracy of the pursuit movements was poor. The proportion of smooth pursuit tracking increased, of course, with age. Typically, 10 to 12-week-olds could catch up to (within several cycles) and track the target at 20 degrees/sec., but could not track the target at 40 degrees/sec. The lack of effective smooth pursuit early in infancy is probably not due to the lower acuity and contrast sensitivity in the same period. Adults who view targets that are optically degraded in
comparable ways do not show reduced accuracy in smooth pursuit. Aslin also reported the onset of what he regarded as predictive tracking in some of the 10 week olds.21

**Hypotheses amenable to simulation (set 7)**

Aslin (1987) has speculated that smooth pursuit eye movements may develop out of saccadic eye movements. Presumably, the later development of smooth pursuit suggests this possibility. However, for this to be true, one might expect early smooth pursuit movements to look more like numerous, but very brief, saccades. On the other hand, it could very well be that the smooth pursuit mechanism is dependent only on an accurate and fast saccade in order to get it started. We will not have time to pursue this topic with simulations. However, it seems that the question of whether smooth pursuit develops out of a saccade mechanism could be explored in simulation with our system. In chapter 5, we discuss some saccade learning experiments that begin to show a jump-like tracking of a moving target. However, more work will be needed to determine whether there is a way that smooth pursuit can evolve out of these beginnings.

**Adult comparison**

For adults, visual acuity is known to be high during pursuit eye movements, which keep the target foveated. Acuity for infants during smooth pursuit, when they are able to do it, has not been determined. Pursuit movements in adults can be as high as 100 degrees/sec., and the pursuit velocity is not directly under voluntary control. Instead, it is determined by the target velocity. The onset latency for pursuit eye movements is 0.13 seconds (Kandel and Schwarts, 1985)

**2.3.4 Selected chronology of developmental events in first few postnatal months**

A brief chronology of the milestones relevant to the development of visual attention in the first few months after birth will provide a context for the visual grasp simulations. We remind the reader that our simulations are only capable of learning to make saccades to stationary and moving targets. The developmental context is provided for a comparison with the development of reaching. The data for the chronology is taken from a variety of sources, including (White, 1970), (White and Held, 1966), (Hofsten, 1990; 1991), (Gruber and Voneche, 1977), (Spelke, 1990), (Aslin, 1987), (Nelson and Horowitz, 1987).

Overall, at birth, there is a rudimentary ocular centralizing reflex, but infants do not show any clear ability to orient to visible targets. However, at 6 months an infant can pursue visual targets at various distances, and within a moderate range of speeds. In addition, he can reach for objects swiftly and accurately. Similarly, an infant is unable to accommodate to visual objects at birth, but by 6 months, (s)he can do so accurately. Soon after birth, infants are able to move their eyes toward flashes of light, and show some

21 We will address the issue of whether eye movements that lead a target should be regarded as "predictive" when we discuss the mechanisms for learning to track moving targets in several parts of subsequent chapters.
evidence of motion perception. In general, infants show a rapid development of various aspects of motion perception. A more detailed chronology of about the first 1/2 year follows.

The first month of life scarcely shows external evidence of significant development.

(a) There is very little waking activity at about 5% of daylight hours.
(b) Infants' activity is mostly that of eye movements.
(c) Nearly all eye movements are saccadic.
(d) Vertical OKN can be measured from birth.
(e) Saccades are fairly accurate in direction from birth onward.
(f) The magnitude of individual saccades tends to be low.
(g) Smooth pursuit eye movements are non-existent within the first month.
(h) Their eyes are locked at a focal distance of about 7.5 inches on average.
(i) If shown a large, contrasty triangle, one month olds will tend to fix their eyes on a single vertex.
(j) If shown a picture of a face, they will either not look at it, or will fix their attention at some small area on the perimeter.

During the second month, there is more noticeable development.

(a) Infants still have a tonic neck reflex posture that precludes mutual explorations by the hands.
(b) Visual accommodation, though limited, begins in the middle of 2nd month, with a range of 8-10 inches.
(c) When a visual target approaches too close, say 4-6 inches, the infant will try to put distance between it and himself by rearing backward, if he can, and turning his head away from the object. This reflex is especially prominent in the second month, and it is present before stereopsis develops.
(d) Eye movements are still almost entirely saccadic. Saccades are still short and many are required to foveate a target. Saccades to peripheral targets are relatively infrequent.
(e) Horizontal OKN improves slightly.
(f) The first, infrequent smooth pursuit eye movements may appear.
(g) When shown a large, high-contrast triangle, some 2 month infants will scan 2 or 3 verticies, while more developmentally advanced infants will also follow the contour of the triangle.
(h) When shown a picture of a face, these infants will scan within the perimeter of the face. How the parts of the face are arranged within the perimeter does not matter to these infants.
(i) Some, but not all, infants at this age can track a slowly moving, brightly colored block with both eye and head movements. Infants who cannot do this task usually continue to foveate the spot where the object initially was when it got their attention.

---

22 The aversive response to a visual cliff is believed to be a kinetic depth effect.
By the third month, infants show more intentional behavior.
(a) Hand regard is a surprisingly frequent activity, at which most infants spend dozens of hours. Visual attention increases sharply at the beginning of the 3rd month along with hand regard.
(b) Tactile exploration with the hands is minimal before the third month.
(c) The blink response to an approaching target also shows a sharp increase at the beginning of the third month.
(d) Color vision is well developed.
(e) During the 3rd and 4th months most infants become visually alert, and it is easy to capture their visual attention, especially with moving objects with sharp corners and high contrast against background (e.g. silvery, shiny, or bright colors, etc.).
(f) Saccades still dominate eye movements, and many are required for foveation.
(g) Horizontal OKN still in transition.
(h) Smooth pursuit eye movements become effective for slowly moving objects.

In the fourth month, perceptual competence begins to show indications that it is becoming more adult-like.
(a) Adult-like accommodation is achieved.
(b) Stereopsis develops rapidly. At 3.5 months, infants can track a moving virtual object specified by binocular disparity in a dynamic random dot stereogram.
(c) The pattern of multiple saccades persists in the fourth month.
(d) OKN will not reach adult horizontal symmetry until the 5th month.
(e) Smooth pursuit improves, but far short of adult speeds.
(f) Up to and including the 4th month, an enriched visual environment is shunned by infants, they seem to prefer a visually quiet environment for early learning. That is, an excessively rich visual environment (up to and including the fourth month) will make infants irritable and reduces their eye explorations.
(g) When infants are preoccupied with an interesting visual object, peripheral events do not distract them as much as they normally would.
(h) It should be mentioned that visual acuity and contrast sensitivity are still well below adult levels at this time.
(i) Infants at this age prefer to look at normal rather than rearranged faces.
(j) At four months, motion can specify object unity, while no static configurational properties can. That is, two adjacent, motionless objects are not perceived as distinct if differing in color, texture or shape. Furthermore, the sensitivity to various static configurational information is not fully developed until 2 years of age!
(k) At this age, all infants can track a slowly moving, brightly colored block with both eye and head movements.

Finally, in the fifth month, infants begin to be able to actively explore their immediate surroundings.
(a) It is only after 1-2 months of sustained hand regard that infants begin to actively explore their environments.
(b) At this time, the richer the visual environment is, the more active are the infant's eye-movements.
(c) At 5 to 6 months infants can generalize from one pose of a face to another.
(d) Saccades still show multiplicity at 5 months.
(e) OKN is at near-adult levels.
(f) Smooth pursuit still not at adult speeds.

There are several patterns in these data that we would like to point out, since they suggest directions for research into the development of visual attention. First, the low acuity, low contrast sensitivity, the short range of visual accommodation and the preference for a quiet visual environment together suggest that infants see a highly filtered image, which may be to good advantage in separating nearby visual objects from background. This separation may have a purpose, namely to bootstrap visual recognition and attention learning mechanisms. All of these features of visual systems early in development may provide simple targets for eye-head and eye-head-hand coordination to develop, before object recognition is mature.

Second, hands and faces are probably the first objects that are scrutinized carefully by infants. Presumably, being able to identify and track each of these categories of objects is extremely useful both for the development of sensorimotor coordination, but also for the development of imitation, which occurs much later in development.

Third, the earliest precursors of attention may be those processes which bring the eye to foveate something moving within the visual field, or to foveate a contrasty corner. These two attributes are the first visual "features" that draw the eye to a location in the visual field.

2.4 Summary

In this chapter we reviewed selected papers from the infant experimental literature on the development of some of the most basic skills involving visuo-motor coordination: eye movements, and visually directed arm movements. Where it was possible, we noted open questions, or conflicting experimental results that we believe can be either resolved or illuminated by simulation. The areas that we identified as amenable to simulation included (1) reflexes as bootstrap mechanisms of the learning apparatus, (2) learning during growth, (3) task-dependent differences in the structure and development of movement unit sequences, (4) task-dependent differences in the development of linearity of movements, (5) reasons for the emergence of a speed-curvature relationship, (6) the minimum senses and motors required to learn the dynamics of reaching, (7) the use of vision in reaching and how it may develop, (8) possible reasons for multiple saccade sequences in infancy, (9) the developmental relationship between saccades and smooth pursuit.

The diverse studies and results reviewed in this chapter suggest that humans, even as infants, adapt to experimental task demands with a quickly learned strategy for performing the task. Such strategies may be subtly selected to match experimental condition and the current abilities of the infant/subject. Thus, it may very well be that the differences that have been observed by infant researchers have resulted from this.
The computational model that will be presented in chapters 3 and 4 has been constructed to address this issue of different learning strategies, and how they may come about from the different task demands. Chapter 5 will present the results of applying this computational model under specific initial conditions to approximate the developmental questions posed in this chapter.
Chapter 3  The Base Level of the Interactive Architecture

3.1 Introduction, motivation, and preliminaries

3.1.1 Introduction

The primary purpose of this chapter is to describe the base level of the unsupervised learning algorithm that we have applied to a variety of learning tasks. These tasks fall within two general categories: visual grasp, and the reaching phase of tactile grasp. The tasks will be described in section 3.2.

In all task applications of the algorithm, the same implementation has been employed, despite the variety of task differences. The only differences in each of the task applications of the algorithm are the senses and motors used (whether simulated or provided by hardware), the routines to compute a collection of cost functionals to be minimized, and an optional, supplied or learnable practise strategy for each task. In chapter 4, we will describe an algorithm by which the system can learn practise strategies.

A second purpose of this chapter is to keep the connections between the theoretical issues, raised in chapter 1, and the architectural choices and features, presented in this chapter, as clear as possible. The next few subsections should help in this regard. The importance of keeping these connections clear cannot be overstated. The algorithms we will present in this and the next chapter do not exactly fit any popular model. As the next subsections will show, the reasons for the differences come from the theoretical context encompassing the work.

3.1.2 Theoretical Issues motivating the Interactive architecture

The form of this learning algorithm has been directly motivated by the developmental meta-theory presented in Chapter 1. We have tried to capture the essential aspects of this developmental meta-theory, and where the theory is incomplete, to extend it in directions consistent with the epistemological orientation adopted in chapter 1. This orientation assumes that biological organisms learn primarily in active ways, through trial-and-error interactions with their environment. What we mean by "active" is that the organism's activity is initiated by internal criteria, and that further activity is guided by the results of its actions in the environment. The results of its actions, though occurring within the environment, are interpreted by internal criteria of the organism. These internal criteria we hypothesize are affect mechanisms. Thus, adaptation is an alteration of an internal program for interacting with the environment. Whether an alteration is regarded as adaptive or not is determined by the organism's goals and affect mechanisms. Thus, activity is internally initiated, internally generated, and corrected by internal criteria that interpret the effects of recent interactions between the organism and environment (in mutual adaptation). None of this presupposes a "representation" of the outside world, at least not in the usual sense of the term.1 We are, in fact, taking a radical position by rejecting the commonly accepted notion of representation. We are hypothesizing that rather than trying to copy or simulate the outside world, the organism learns to interact

---

1 However, we are assuming that evolution has provided the organism with developmental mechanisms, whose structure may be quite complex.
with the world for its own ends and with its own means. These programs for interaction are the internal structures that get built by the organism's developmental mechanisms (as opposed to representations).²

Thus, our direction in constructing the algorithms of this chapter and the next comes from a desire to build an interactive system with a bit more self-guidance and flexibility in its ability to find a solution to a task, than other methods may provide. This includes the beginnings of an ability to learn to train itself.

Lastly, having posed the problem of development as the problem of constructing effective interactions with an environment, we have sought to learn about development, so defined, through the trial-and-error construction of an interactive system.

### 3.1.3 Implications for the design of the learning architecture

Before proceeding with a description of the bottom level learning system, it seems appropriate to state some of the motivating guidelines. These guidelines are consequences, in some cases, of the theoretical position taken in chapter 1 and repeated in the previous section. In other cases, they are consequences of the developmental experimental results presented in chapter 2.

1. The system should learn by self-generated, trial-and-error, interactions with its environment.³
2. The consequences of an interaction are assessed by the learning system itself, internally by the computational analog of an affect mechanism.
3. The system should be goal-oriented in the sense discussed in chapter 1.⁴
4. The internal representations that the system creates and uses have more to do with phases or states of the interactions, than with a model of the states of the world. In fact, no attempt is made to "model the world" explicitly.
5. Also wanted is the ability to store any typical reaching trajectory, rather than having only a 1 trajectory system. This forces certain economies in the database design. Of course, the system should be able to retrieve a trajectory and execute it in real time.
6. We wanted to explore ways in which attention mechanisms could "evolve from" the goal-oriented bottom level system. This topic will be treated in the next chapter.
7. The system should have some ability to work with multiple cost functionals. Perhaps, with some ability to learn by attention to specific cost functionals, or combinations of cost functionals.

---

² We believe that this is the essence of Piaget's theoretical position w.r.t. the mechanisms of development.
³ Most of what kids learn, according to Piaget, is learned with remarkably little parental intervention or instruction.
⁴ In the process of doing away with copy notions of knowledge as well as passive, externally driven organisms, the externalized notion of goal can also be discarded. To replace it, a distinction must be made between the internal regulatory definition of goal and the external target that is sometimes the object of the goal-driven activity. Within such a framework the notion of representation is also unnecessary.
(8) There should be some economy in the learning process so that the entire state space of possible movements between two targets would not need to be searched. This complements another requirement, which is that the system not represent states it has not visited. Thus, it does not attempt to represent or search all possible states of the world or of the system. Instead, when it finds a trajectory that is poor, but works, it searches near it for improvements.

(9) The system should learn opportunistically as events in the environment present themselves (unsupervised in a stronger than usual sense). However, it can also initiate practise even when there is no obvious target.

(10) The system should have some added flexibility to be able to get itself unstuck from local minima, if its current search algorithm does not appear to be working.

(11) As we assume is true of humans every movement attempt is an opportunity for learning. There are distinctly separate practise and execution phases, though the system can choose to "practise" when there are no higher priority goals active. However, such practise sessions use precisely the same learning algorithm as the usual learning/interacting process.

(12) The learning system should be able to bootstrap itself from basic reflexes, without a desired trajectory, etc.

(13) The same architecture should work for all tasks, with modular replacement of senses, motors, and cost functionals, etc.

(14) At the level of action, we need a "representation" that allows disparate elements, so that the system could piece together a movement with more than one type of sense-action element.

(15) Rather than explicitly discretizing the possible states of the world/system, the preference was to use variables whose values were taken from bounded, continuous domains, together with interpolation to fill more of the space in the vicinity of the samplings.

(16) We sought clear parallels with basic notions from Piaget's theory, such as adaptation, assimilation, accommodation, functional assimilation, and equilibration for starters.

(17) As with biological systems, there is no artificial separation between practise and action.

3.1.4 Developmental preliminaries

In order to keep the parallels clear between developmental theory, and our learning algorithm, we will describe the algorithm using terms whose psychological meanings are due to Piaget, and whose algorithmic meaning will be given in this chapter. First, we will define some basic notions as they have been used by Piaget (e.g. Ginsburg and Opper, 1969).

For Piaget, adaptation is the ability to achieve goals in the face of a changing environment. This involves interacting with the environment, with particular emphasis on the organism's ability to initiate and guide activity. Processes that determine which activity to use are referred to as goals. The activity that ensues when a goal has become active is a scheme, and the program for interaction that produces the scheme is referred to as a schema. Finally, adaptation is considered to be composed of two complementary
processes: **assimilation** and **accommodation**. Assimilation is the process of choosing which schema (i.e. program for interaction) to activate, and accommodation is the altering of the schema so that it works in a new situation.

In an attempt to clarify these definitions, and to make them more precise, we will consider them within the context of making saccadic eye movements. The notion of goal may be somewhat clear. For example, the goal of "foveating a target in the peripheral field of view by making a saccade" is a fairly low level goal, associated, in this case, with a topographic map within the superior colliculus. One question that immediately arises is, what is the schema here? That is, where and what is the program for making saccadic eye movements (i.e. the schema for making eye movements). Is it the entire topographic map, or is it a subset of cells within the topographic map associated with making a particular saccade to a particular target? We have chosen to refer to the former case as the schema or **schema class**, and to refer to the latter case as a **schema instance**. In this terminology, the topographic map associated with the goal of making a saccadic eye movement contains the schema (class), which in turn is a collection of schema instances for particular movements to particular targets.\(^5\)

Within this context we can clarify the notions of assimilation and accommodation. When the goal of making a saccade is **enabled or active**, and there is a target available, then the location of the target is used as an index into the database of possible saccades, the topographic map, and a schema instance is retrieved. This indexed retrieval is the process of assimilation.\(^6\) Furthermore, if the schema instance does not move the eye to foveate the target as expected, then it needs to be modified. The modification process is a process of accommodation.

There are two additional notions from Piaget that can be introduced here, namely **functional assimilation** and **generalizing assimilation**. Functional assimilation is a tendency for a "cognitive structure" (i.e. a program for interaction), in this case a schema, to be exercised (i.e. executed as a program) even when there may be no suitable object of interaction (a substitute object may be found, or sometimes the activity occurs without an object, like pantomime). For example, there are reaching movements that are made to grasp and retrieve an object, but there are reaching movements made purely for practise. We have interpreted functional assimilation specifically as a mechanism for practising schema instances within a topographic map that have been used (hence practised) less often than others. Because of infrequent use, they require execution with a substitute object, or no object, in order to better approximate the average level of performance across the topographic map.\(^7\) On the other hand, generalizing assimilation is the tendency of a schema (or perhaps schema instance) to be applicable to wider and wider contexts. In

---

\(^5\) Since it is cumbersome to use the term schema instance, we will often refer to a schema instance with the term schema, when the context makes it clear which we mean.

\(^6\) It should be mentioned that there are two components to the index: the location of the target, and the goal itself.

\(^7\) We have implemented functional assimilation as an independent goal process, with its own priority (usually low) and mechanism for assessing its effectiveness.
our system, the use of approximation techniques produces, as a side-effect, a kind of 
generalizing assimilation.

3.1.5 Approximation theory preliminaries

In this section we will review some introductory material on approximation theory 
and discuss how it was used in the thesis. More complete treatments of approximation 
theory, applications of approximation theory, and RBF approximation methods and their 
relation to either regularization theory or to neural networks can be found among the 

Suppose that \( f \) is a multi-variable, vector-valued, continuous function that we can 
sample, but do not know. That is, we do not have an analytic expression for \( f \). Specifically, \( f \) is a map

\[ f : X \rightarrow Y \]

where

\[ X \subseteq \mathbb{R}^n, Y \subseteq \mathbb{R}^m \]

for some integers \( n \) and \( m \). Suppose also, that we have a sample, \( D \), of "input-output" 
pairs of \( f \). That is,

\[ D = \{ (\bar{x}_i, \bar{y}_i) \in X \times Y \}_{i=1}^N \]

where

\[ f(\bar{x}_i) = \bar{y}_i, i = 1, \ldots, N \]

The problem of approximating or interpolating \( f \) is usually expressed as one of finding an 
approximating function,

\[ F(W, \bar{x}) \]

where \( W \) is a vector of \( k \) parameters from some set \( P \). This amounts to picking \( F \) and 
finding the set of parameters, \( W \), for which \( F \) "comes closest to" \( f \). That is, for a norm, 
\( \| \cdot \| \), on the appropriate function space,

\[ ||f(\bar{x}) - F(W, \bar{x})|| < \varepsilon \]

and for an adequately small distance, \( \varepsilon \). Furthermore, a best approximation is one for 
which

\[ ||f(\bar{x}) - F(W, \bar{x})|| \leq ||f(\bar{x}) - F(W', \bar{x})|| \forall W' \in P \]

In other words, \( W \) is the parameter set from \( P \) for which \( F \) is closest to \( f \).

From either the neural net point of view, or the approximation theory point of view 
(Rumelhart and McClelland, 1986), the learning step is usually regarded as this step of 
finding the best selection of parameters, \( W \), having already selected \( F \). In our learning 
algorithm, on the other hand, the task of finding a "good" example set, \( D \), has been given
at least as much attention as the more "traditional" learning step of finding the parameters of the approximator. In fact, a developmental perspective necessitates a broader view of what phenomena we chose to tag with the term "learning."

At this point, we will summarize the connection between regularization theory and approximation theory as discussed in (Poggio and Girosi, 1989). This discussion will ground and motivate the approximation techniques we have implemented for the schema database.

For this discussion, assume that we are sampling a real-valued, multi-variable function, which we wish to approximate with a function, \( f^1 \) (note that \( f^1 \) does not correspond to \( f \) of the previous discussion, but could correspond to a 1-dimensional component of \( F \)). Similarly, the sample set will be denoted

\[
D^1 = \{(\bar{x}_i, y_i) \in X \times Y \}_{i=1}^N
\]

where

\[
X \subseteq \mathbb{R}^a, \quad Y \subseteq \mathbb{R}^b.
\]

If we are treating the problem as an interpolation problem, then we have

\[
f^1(\bar{x}_i) = y_i, \quad i = 1, \ldots, N.
\]

Otherwise, we do not assume that our approximator goes through the data points. In regularization theory, a technique for finding \( f^1 \) is to minimize a cost functional of the following form (see Poggio and Girosi, 1989)

\[
H(f^1) = \sum_{i=1}^{N} (y_i - f^1(\bar{x}_i))^2 + \lambda ||Pf^1||^2
\]  

(3.0.a)

The summation term measures the distance between the sample data and the solution, \( f^1. \) The second term is referred to as the stabilizer. It measures the cost associated with deviating from a constraint (e.g. \( C_{\text{jerk}} \) in expression 3.1.3 in section 3.2.2 of this chapter). \( P \) is a linear differential operator in this formulation of the regularization principle, \( ||.|| \) is the \( L_2 \) norm (see Royden, 1968; Rudin, 1973) and \( \lambda \), the regularization parameter, is a real number in \( \mathbb{R}^+ \), which balances the tradeoff between the summation term (often an error) and the constraint term (often a smoothness constraint). We have used \( \lambda = 1 \) in our computations.

As shown in (Poggio and Girosi, 1989) the solution to 3.0.a has the form
\[ f^1(\bar{x}) = \sum_{i=1}^{N} c_i G(\bar{x}; \bar{x}_i) \]

(3.0.b)

where \( G(x;x_i) \) is the Green's function of the differential operator \( \text{adjoint}(P)P \) centered on \( x_i \).\(^8\) Furthermore, the coefficients, \( c_i \), can be computed from the N equations,

\[ c_i = (y_i - f^1(\bar{x}_i))/\lambda , \]

together with 3.0.b, evaluated at the N data points. This is equivalent to solving the following matrix equation,

\[(G + \lambda I)\bar{c} = \bar{y}, \]

where

\[ (\bar{y}) = y_i , \quad \bar{c} = c_i , \quad (G)_{ij} = G(\bar{x}_i; \bar{x}_j) \]

Equation 3.0.c

For various reasons, including robustness, we inverted the matrix of centers, \( G \), using Singular Value Decomposition. As observed in (Poggio and Girosi, 1989), since \( \text{adjoint}(P)P \) is self-adjoint, its Green's function is symmetric (i.e. \( G(x,y) = G(y,x) \)). Beyond this, if \( P \) is rotationally and translationally invariant, then \( G \) will be a radial function (i.e. \( G = G(||x-y||) \)).\(^9\) Consequently, the solution expressed in 3.0.b is equivalent to a Radial Basis Function method. That is,

\[ f^1(\bar{x}) = \sum_{i=1}^{N} c_i G(||\bar{x} - \bar{x}_i||) \]

where \( G \) can be a Gaussian,

\[ G(x) = e^{-\|x\|^2/\sigma^2} \]

Thus, the solution of the regularized problem has the following simple form, where we have again omitted the polynomial terms.

---

\(^8\) It should be mentioned that we are ignoring the null space of the operator, \( P \), and any additional terms that it may contribute to 3.10.2. In particular, if the Green's function is conditionally positive definite of some order, then the null space of \( P \) is given by a sum of polynomials.

\(^9\) In the cases considered by this thesis, we are assuming that \( P \) is rotationally and translationally invariant.
\[ f^1(\vec{x}) = \sum_{i=1}^{N} c_i e^{-\| x - x_i \| / \sigma} \]

This concludes our discussion of the approximation methods used in our simulations. We will discuss how these approximation methods were integrated into the learning algorithm later in this chapter.

### 3.1.6 Section summary

The comments of the preceding sections have identified three different aspects of learning. In the section on approximation theory, the learning step was identified as the process of finding the parameter set that specifies the approximation. Another way to look at this is that the learning step is finding a smooth surface that "fits" the data according to some criterion.

Second, and also in the section on approximation theory, we identified the acquisition of an example set as another aspect of learning. Acquisition is a potentially misleading term, for as we will see later in this chapter and next, constructing a good example set involves an active, trial-and-error process of continually absorbing elements into the example set and discarding elements from it. Thus, the example set is not a static thing to be acquired and forgotten about. In addition, the example set has structure to it beyond the structure associated with the mathematical notion of set. This too will be discussed later in the chapter.

Third, there is the notion, quite explicit in Piaget, that learning involves structural change. This provokes the questions, "what are the structures and what are the mechanisms of change?" Our answer to the first question has been that the structures are control structures for interacting with an environment. Potential candidates for these structures will be discussed later in this chapter. An answer to the second question includes goal processes, affect mechanisms and other hypothetical mechanisms presented in this chapter and the next.

### 3.2 Task applications of the architecture

#### 3.2.1 Arm movement task description

In the first category of tasks, the learning system is "connected" to a simulated robot arm, and learns to make planar reaching movements to objects as they appear within its field of view. Objects can be moving or stationary, though most often they have been stationary. The learning system can monitor the position of the object in time, and will continue to make practise movements until it can reliably hit the target, or until the target disappears. After it has succeeded at contacting the target, it will continue to practise, in order to improve accuracy and smoothness of the movement. Targets appear at various locations and with varying durations, essentially at random from the point of view of the learning system.
Figure 3.1: The simulated arm is a 2-joint, planar, dynamic arm. $\theta_1$ and $\theta_2$ were measured as shown with clockwise rotations of the joints corresponding to more positive joint angles. The lengths and widths of the links can grow at different, and variable rates, which changes the masses and moments of inertia of the links. Friction can be set for either joint, and stiffness can be set for either actuator (as the actuator gain). Joint movement is restricted as follows, $-180 \leq \theta_1 \leq 0$ and $-135 \leq \theta_2 \leq 0$. 
Figure 3.1 depicts the simulated robot arm, which has 2 links and 2 joints. The range of possible joint angles is restricted to a range of 180 degrees for the shoulder joint, $\theta_1$, and 135 degrees for the elbow joint, $\theta_2$. Due to the 2-dimensional workspace of the arm, the visual perspective assumed by the simulation is that of viewing the workspace from directly overhead. Targets may appear within the 2-d workspace as either stationary or moving. When moving, they are given linear trajectories and move at a constant velocity.

The links and joints of the robot arm have characteristics that can be varied in time, independently of the learning system. The lengths and thicknesses of the links can grow, which has the side effect of increasing the mass and altering the inertia tensor of the arm. Similarly, the joints have viscous and Coulomb friction that can be non-zero, if desired, and can be made to change with time. The dynamics of the robot arm are simulated using an iterative Newton-Euler dynamic algorithm similar to that presented in (Craig, 1986).

The learning system does not have access to any of these parameters. Instead, the learning algorithm can sample the angular position and velocity for each link, the position and force of any tactile contact with either link, and a pain variable for contact that exceeds a threshold of force. In addition, the visual position and velocity of any present object(s) is available. A visual image of the hand is also available, though it has not been used in our simulations. A visual image of the arm is not assumed because we are modeling the learning process both early in development, and we are modeling the reaching process, prior to grasping. Future versions of the simulation will address the problem of how an organism can learn to interpret and use the visual image of its own limbs.

Control of the arm can be selected from among several controller modules. The learning system can select from among these controllers within or between movements. There are modules for applying a specific torque vs. time function that can be translated to the desired time interval and scaled to a desired amplitude. Another module servos the arm to an equilibrium position during a specified time interval. The equilibrium position and the time interval over which the servo control occurs are specified, of course, by the learning system.

3.2.2 Arm movement task stated mathematically

There are several ways to frame the learning problem of the arm task. We will explore some of the alternatives in this section. Suppose the articulator has N dimensions, then the vector of torques delivered by the actuator at time, t, is denoted

---

10 This may seem like a trivialization, but some infant reaching tasks are remarkably close to this, especially since depth perception is developed at by the time that reaching movements become effective. However, future versions will operate in 3-d. There is nothing inherent in the system that limits it to 2-d.

11 The equilibrium position controller for the arm task simulates a spring-like torque to the commanded position, but also has a damping mechanism that is linear with velocity. However, we found an additional term supplying a quadratic torque restoring force to be more effective than just the linear, spring-like term.
\[ \ddot{\tau}_i = (\tau_1, \tau_2, \ldots, \tau_N)_i. \]

If some other action is to be taken, rather than applying torques, then the corresponding action vector is denoted

\[ \ddot{a}_i = (a_1, a_2, \ldots, a_N)_i. \]

Following the usual vector conventions for the position and velocity of the target, and the position and velocity of the articulator, we have

\[ \ddot{X}_{\text{target}} = (x, y, z)_{\text{target}}, \]

\[ \ddot{V}_{\text{target}} = (\dot{x}, \dot{y}, \dot{z})_{\text{target}}, \]

\[ \ddot{\theta}_{\text{articulator}} = (\theta_1, \theta_2, \ldots, \theta_N)_{\text{articulator}}, \]

\[ \ddot{\dot{\theta}}_{\text{articulator}} = (\ddot{\theta}_1, \ddot{\theta}_2, \ldots, \ddot{\theta}_N)_{\text{articulator}}. \]

Then the arm task is one of finding a function, \( T_{\text{map}} \), which has the following form\(^\text{12} \),

Equation 3.1

\[ \ddot{\tau}_i = T_{\text{map}} (\ddot{X}_{\text{target}}, \ddot{V}_{\text{target}}, \ddot{\theta}_{\text{articulator}}, \ddot{\dot{\theta}}_{\text{articulator}}, t) \]

which we have chosen in preference to the more usual form of the problem. The alternative, and more common formulation, involves comparing the position and velocity of the hand in visual space with the position and velocity of a "desired" trajectory. This form of the task can be expressed as,

\[ \ddot{X}_{\text{hand}} = (x, y, z)_{\text{hand}}, \text{ and} \]

\[ \ddot{V}_{\text{hand}} = (\dot{x}, \dot{y}, \dot{z})_{\text{hand}}, \text{ yielding the alternative form}, \]

\[ \ddot{\tau}_i = T_{\text{map}} (\ddot{X}_{\text{desired-hand-trajectory}}(t), \ddot{V}_{\text{desired-hand-trajectory}}(t), \ddot{X}_{\text{hand}}, \ddot{V}_{\text{hand}}, \ddot{\theta}_{\text{articulator}}, \ddot{\dot{\theta}}_{\text{articulator}}, t) \]

Equation 3.2

\(^{12}\) Though we have included a target velocity term, our data is mostly collected from the stationary target version of the task. However, the learning system is capable of learning to reach toward moving targets, by reaching a little ahead (no planning is needed to learn this as we will discuss later) and matching the speed of the target.
Typical constraints on the problem as framed in Equation 3.1 may include minimizing a subset of the following cost functionals. Clearly, the hand has to end up near the target. In some formulations the location of the hand is determined visually, yielding

\[ \text{Cost}_{\text{norm}} = |\vec{X}_{\text{hand}} - \vec{X}_{\text{target}}| \quad \text{at} \quad t = t_{\text{final}}. \]  

(3.2.1)

In our formulation, the final position of the hand in relation to the target is determined by the location on the arm or hand where contact is made. The corresponding cost functional is

\[ \text{Cost}_{\text{norm}} = \frac{|\vec{X}_{\text{hand}} - \vec{X}_{\text{contact, site}}|}{|\vec{X}_{\text{hand}} - \vec{X}_{\text{shoulder}}|} \quad \text{at} \quad t = t_{\text{final}}. \]  

(3.1.1)

Since we want the hand to hit the target with as little force as possible, we can either use a functional that measures the angular speed of the links of the arm, as below.

\[ \text{Cost}_{\text{norm}} = \sqrt{\dot{\theta}_{\text{link, 1}}^2 + \dot{\theta}_{\text{link, 2}}^2} \quad \text{at} \quad t = t_{\text{final}}. \]  

(3.1.2)

Other possibilities include using force sensing, or a combination. More important are constraints on the accumulated jerk of the entire reaching movement (jerk is the derivative of acceleration).

\[ \text{Cost}_{\text{Jerk}} = \frac{1}{2} \int \left( \frac{d^3 x_{\text{hand}}}{dt^3} \right)^2 + \left( \frac{d^3 y_{\text{hand}}}{dt^3} \right)^2 \, dt \]  

(3.1.3)

An alternative is to measure the accumulated torque change of the movement, which can be argued to be equivalent to the minimum-jerk functional (Kawato, et al, 1987; 1988; 1990).

\[ \text{Cost}_{\text{torque}} = \frac{1}{2} \sum_{i \in \text{arm}} \left( \frac{d \tau_i}{dt} \right)^2 \, dt \]  

(3.1.4)

Even with some combination of these functionals, such as the final position, final velocity and minimum jerk functionals, \( T_{\text{map}} \) is severely underconstrained. For example, nothing in the above costs explicitly encourages the mean velocity for the movement, although the minimum jerk constraint by itself will tend to drift toward slower movements over time. However, this problem can be remedied, for example, with a cost functional that minimizes cost for either a particular duration for the movement, or a particular maximum velocity within the movement. Furthermore, the minimum-jerk constraint tends to induce a simple one-peak velocity profile. Nevertheless, it is clear that ordinarily there is no unique solution to \( T_{\text{map}} \) under these constraints. In general unique solutions in the
case of more than two links are not possible.\footnote{With two links solutions come in pairs, one of which we have eliminated by bounding the elbow and shoulder joints at angles comparable to human bounds, so that the elbow out and elbow in versions do not both occur.} However, the learning architecture that will be presented later in this chapter and the next tends to build on the particular solution it discovers first. Though it is still possible for a system such as the one presented in this chapter to cycle between two different solutions, rather than improve one solution, the additional mechanisms of chapter 4 can detect and correct for such occurrences.

These considerations taken together are, in part, why a more common approach to the problem is to assume that the position and velocity profile of the hand (at all time steps) during the movement is known. This is the approach taken in Equation 3.2. In this case, the costs to be minimized are typically sums of squared errors, where the sum is taken for the whole movement. The continuous formulation for position and velocity errors is given below.

\[
\begin{align*}
\text{Cost}_{\text{position-error}} &= \frac{1}{2} \left[ \int_0^t \left( (x_{\text{hand}}(t) - x_{\text{desired-trajectory}}(t))^2 + (y_{\text{hand}}(t) - y_{\text{desired-trajectory}}(t))^2 + (z_{\text{hand}}(t) - z_{\text{desired-trajectory}}(t))^2 \right) dt \right]^{1/2} \\
\text{Cost}_{\text{velocity-error}} &= \frac{1}{2} \left[ \int_0^t \left( (x_{\text{hand}}(t) - x_{\text{desired-trajectory}}(t))^2 + (y_{\text{hand}}(t) - y_{\text{desired-trajectory}}(t))^2 + (z_{\text{hand}}(t) - z_{\text{desired-trajectory}}(t))^2 \right) dt \right]^{1/2}
\end{align*}
\]

(3.2.2)

There are a number of ways to "solve" this version of the learning problem. However, any method will take advantage of the error terms that are available at each time step. Since it is necessary to minimize the following discretized cost functional (as in any approximation problem) with each practice attempt and for each time step \( i \) from 0 to \( m \).

\[
\sum_{i=0}^m \left\| T_i^{\text{desired}} - T_i^{\text{map}}(\mathcal{X}_i^{\text{target}}, \mathcal{V}_i^{\text{target}}, \mathcal{X}_i^{\text{hand}}, \mathcal{V}_i^{\text{hand}}, \mathcal{\bar{b}}_i^{\text{articulator}}, \mathcal{B}_i^{\text{articulator}}) \right\| + \text{stabilizer}
\]

(3.2.3)

For simplicity we will ignore the stabilizer term that specifies the "smoothness" of the function being approximated. Observe that the position and velocity errors can be used to generate feedback torque corrections from the position and velocity errors. That is, for each \( i \)
\[ T'_{\text{feedback-corr}} = -\tilde{K}_{\text{position}} \cdot (\tilde{X}'_{\text{desired}} - \tilde{X}'_{\text{attempted}}) - \tilde{K}_{\text{velocity}} \cdot (\tilde{V}'_{\text{desired}} - \tilde{V}'_{\text{attempted}}) \]  

(3.2.4)

On the next movement the torque to try is the current approximated surface plus a feedback correction,

\[ T'_{\text{attempt}} = T'_{\text{approximated}} + T'_{\text{feedback-corr}} \]

for each \( i \). The feedback corrections are essentially an application of Newton’s method, and assuming nothing bizarre happens with the evolving approximated surface, the feedback corrections needed should converge to zero. As more movement attempts are made the \( i \)th difference between desired and attempted torques goes to zero.

\[ \left\| T'_{\text{desired}} - T'_{\text{attempt}} \right\|_{\text{attempts}} \to 0 \]

Consequently,

\[ \sum_{i=0}^{m} \left\| T'_{\text{desired}} - T'_{\text{attempt}} \right\|_{\text{attempts}} \to 0 \]

Several observations should be made regarding such an approach. First, the credit assignment problem is "automatically" eliminated, or assumed to be handled by whatever process computes the visual errors.

Recall from the discussion in chapter 2, that there is no evidence that such information is used by infants as they are learning eye-hand-arm coordination. In fact, the evidence appears to be against this assumption. Some final points should be made to motivate our approach to the problem. It is commonly assumed that the visual system can "imagine" a straight line in space for the trajectory of the hand, and that this trajectory is followed as reaching movements are learned. However, neither evidence nor argument have been put forward that this is the way arm movements are guided (during learning or otherwise), or that the generation of position errors along the trajectory could be produced by the visual system (while solving the necessary perspective transformations and coordinating depth information) within the time required for normal reaching movements.

Nevertheless, one could argue that these difficulties are not as severe as the ill-posed problem that our formulation presents. The disagreement between the two approaches outlined above really comes down to different beliefs about where the difficult part of the problem is. Our view is that human infants solve a very hard learning problem, and we would like to understand the nature of this learning problem. The alternative view implicitly assumes that the learning problem is relatively easy, and the hard part of the problem is "solved", perhaps innately, by the visual system. We would like to suggest that
either to push the difficult part of the problem into vision, or to push it back into phylogeny is a form of avoidance.

To return to our version of the problem, the cost functionals for final position error \(3.1.1\), velocity at contact \(3.1.2\) and accumulated jerk \(3.1.3\) do not provide any error information prior to the end of the arm movement. Thus, computing a function such as

\[
\sum_{i=0}^{m} \| \mathbf{T}_{\text{desired}} - \mathbf{T}_{\text{map}}(\mathbf{\hat{X}}_{\text{target}}, \mathbf{\hat{V}}_{\text{target}}, \mathbf{\hat{\theta}}_{\text{articulator}}, \mathbf{\hat{B}}_{\text{articulator}}) \| + \text{stabilizer}
\]

(3.1.5)

is not a simple matter. For example, \(3.1.1\) and \(3.1.2\) are costs for final position and final velocity errors, which are to be minimized. The initial position and velocity errors are 0 because the system always starts from a known position at rest. Thus, we can compute the first and last terms of the above summation, with no direct way to sample data in between. However, we do know what the stabilizer term should be for this problem. It can be either the accumulated jerk computation \(3.1.3\), or the accumulated torque change computation \(3.1.4\). It is not obvious that the use of either stabilizer is sufficient to provide reasonable solutions to the above minimization problem. Our results, presented in chapter 5, will show that with the appropriate method of search for the "missing terms", a minimization of \(3.1.5\) can be accomplished.

Such a computation, whether accomplished implicitly or explicitly, requires the solution of a temporal credit assignment problem. Much of the remainder of this chapter is a discussion of our unsupervised learning algorithm that solves the temporal credit assignment problem as well as the inverse dynamics of the task.\(^{15}\)

In summary, the "standard" solution to the problem of reaching movements, as expressed in Equation \(3.2\), is well-posed from the point of view of regularization theory. However, it is not clearly supported by studies about how humans learn to make reaching movements. That is, it makes certain unsubstantiated assumptions about the nature of visual processing during reaching movements.

On the other hand, an alternative solution to the problem of reaching, as expressed by Equation \(3.1\), is almost well-posed from the point of view of regularization theory. That is, a very heavy dependence on the stabilizer is required, and some unusual means of finding datapoints is also required. However, this alternative does not make strong assumptions about the nature of the task. Furthermore, our algorithm solves this task without presuming anything undemonstrated about visual system capabilities.

---

\(^{14}\) We have also included a cost associated with the final velocity. The cost is zero when the final velocity is zero, and the cost is monotonically increasing with increasing final velocity.

\(^{15}\) The inverse dynamics problem is well known not to have a closed form solution.
Figure 3.2: The head-eye hardware (and simulation) has four degrees of freedom. In addition to $\theta_{\text{tilt}}$ and $\theta_{\text{pan}}$ each camera can be panned independently, which would support vergence movements or faster horizontal tracking. All actuators are 1000 position stepper motors. The actuators are controlled by microcomputer with 1 step accuracy.
3.2.3 Head-eye movement task description

In the second category of tasks, visual targets would appear anywhere on a 2-d plane in simulation, or within the depth of focus of the MIT vision machine's camera(s), with the position and velocity of the object accessible to the learning system in visual system coordinates.\textsuperscript{16} The learning system would then command practise movements to the simulated or physical head-eye system until it could successfully foveate the target, which was usually moving in this category of tasks. Practise normally continues toward improving accuracy and speed of foveation, as long as the object is in view.

In the case of vision machine use, moving targets could be isolated and selected with a simple algorithm looking at successive differences of digitized frames.\textsuperscript{17} The plane of movement was usually orthogonal, or nearly so, to the axis of the camera lens when all the actuators of the head-eye system were set to their central positions (the origin in articulator coordinates). Targets were moving in linear trajectories and at constant velocity.

As we will discuss below, our simulations did not learn the dynamics as in the arm tasks, but are non-trivial in other ways. Both simulations and vision hardware were equipped with horizontal and vertical movements, controlled by commanding an absolute or relative vertical or horizontal angle (Figure 3.2).

In the case of the MIT vision machine hardware, there are the equivalent of vertical and horizontal head movements, and horizontal eye movements, which are independent for each eye.\textsuperscript{18} In our experiments we only used 2 degrees of freedom, though the restriction was not necessary.

The MIT head-eye system was designed with an electronic controller that servos to specific angular positions, but which cannot be easily or safely circumvented. Consequently, it was not possible for our learning system to learn the dynamics of the head-eye hardware. However, the controller-plus-hardware combination has certain characteristics that make learning to saccade accurately to moving targets non-trivial. First, the slew + settling time of the controller-plus-hardware is non-linear for large angles, and can be somewhat unpredictable. That is, for small angles, such as 1 to 30 degrees, the head-eye will servo to the desired position and stop in roughly linear time. However, as the commanded delta angle gets larger, the slew + settling time becomes larger than if several commands of smaller angles, which sum to the larger angle, had been made. At such large delta-thetas, there would be occassional overshoot and hunting of the system around the desired position. At delta-thetas bigger than 60 degrees, the hunting can be considerable (> 1 minute in duration), and sometimes unstable. The task for our learning system is to make accurate saccades to moving targets, and to find the shortest, reliable

\textsuperscript{16} The head-eye movement tasks we have chosen use either the MIT vision machine hardware, or a computer simulation of the hardware.

\textsuperscript{17} For real-time performance, frames could be dumped directly from the frame buffer to a Connection Machine, and object location and velocity returned for further processing.

\textsuperscript{18} The MIT vision machine hardware is capable of vergence movements, though we did not set up our learning tasks to use them.
time for each movement, despite the fact that the relatively massive head-eye hardware cannot move as fast as a human eye or head.

3.2.4 Head-eye movement task stated mathematically

The mathematical formulation of the head-eye movement task category is similar to that for arm movements. However, one difference is that the head-eye hardware does not permit torque specification as an action. Instead, all actions are commands to tilt or pan head-eye articulators to some absolute angle. Furthermore, the hardware also determines the duration of the servoing to a specified equilibrium angle. The duration of the servo action cannot be preset and can be quite variable for larger angles of movement. The following expressions are notation for the action vector, the target position vector, the target velocity vector, the articulator position vector, and the articulator velocity, respectively.

\[
\ddot{a}_i = (a_1, a_2, ..., a_N), \quad \dot{a}_i = (\dot{a}_1, \dot{a}_2, ..., \dot{a}_N)
\]

\[
\ddot{X}_{\text{target}} = (x_{\text{retina}}, y_{\text{retina}})_{\text{target}} \]

\[
\dot{V}_{\text{target}} = (\dot{x}_{\text{retina}}, \dot{y}_{\text{retina}})_{\text{target}} \]

\[
\ddot{\theta}_{\text{articulator}} = (\dot{\theta}_{\text{pan}}, \dot{\theta}_{\text{tilt}})_{\text{head}} \]

\[
\dot{\theta}_{\text{articulator}} = (\ddot{\theta}_{\text{pan}}, \ddot{\theta}_{\text{tilt}})_{\text{head}} \]

With this notation, then we wish to solve for the function \( T_{\text{map}} \) in Equation 3.3, and which minimizes costs 3.3.1, and 3.3.4, below. The "best" solution under these constraints for the existing hardware is not necessarily a single action, or saccade. Furthermore, even for regions of the \( T_{\text{map}} \) where the best solution is one action, the system can sometimes learn this action most efficiently and safely by using initial trial-and-error sequences of actions of length greater than 1 (usually 4 or 5) and subsequently learning how to reduce the length of the sequence. The learning process for this task will be discussed in more detail in chapter 5.

As a final word about the head-eye problem, notice that Cost 3.3.4 consists of two terms. The right hand term (lower term) is the duration of the process of locating the target and servoing the head-eye hardware as close as possible to that location. An expansion of the right hand term is given as expression (3.3.2). This is the duration that can have a stochastic component, if one of the saccades is very long. The left hand term is a period of time during which the head-eye hardware is stationary, but during which the target may not be. For a fixed period of sampling, the video hardware finds what is probably the target and tracks its movement across the retina. The cost corresponds to how close the target comes to the center of the fovea during this fixed duration sampling period. Using a cost of this type facilitates learning even though it introduces a more complicated pattern of local minima to the cost function. The reason is that it gives a lower cost to head-eye movements that land the gaze in a position where the target
subsequently moves into the fovea center, than the cost of landing the gaze the same distance behind the fovea, with the target moving away from the fovea center.

**Equation 3.3**

\[
\tilde{a}_i = T_{map}(\tilde{X}_{target}, \tilde{V}_{target}, \tilde{\theta}_{articulator}, \dot{\tilde{\theta}}_{articulator}, t)
\]

(3.3.1)

\[
\text{Cost}_{\text{achieve-to-fovea}} = |\tilde{X}_{\text{fovea-center}} - \tilde{X}_{\text{target-nearest-fovea-center}}|
\]

\[
\text{Cost}_{\text{second saccades}} = |t_{\text{end of saccade-sequence}} - t_{\text{target-initially-spotted}}| \\
= \sum_{s \text{ initial saccade}} (|t_{\text{saccade ends (s)}} - t_{\text{saccade begins (s)}}| + |t_{\text{target_position_on_retina_computed (s)}} - t_{\text{head_eye_movement_settles (s)}}|)
\]

(3.3.2)

\[
\text{Cost}_{\text{end saccades}} = |t_{\text{target-nearest-fovea-center}} - t_{\text{end of saccade-sequence}}| \\
= \sum_{r \text{ initial visual sample}} |t_{\text{target_position_on_retina_computed (r)}} - t_{\text{start_sampling_for_target_position (r)}}|
\]

(3.3.3)

\[
\text{Cost}_{\text{end saccades}} = |t_{\text{target-nearest-fovea-center}} - t_{\text{end of saccade-sequence}}| \\
+ |t_{\text{end of saccade-sequence}} - t_{\text{target-initially-spotted}}| \\
= \text{Cost}_{\text{second saccades}} + \text{Cost}_{\text{end saccades}}
\]

(3.3.4)
Figure 3.3: The basic adaptive loop is the smallest unit of adaptive control in the system. The goal and affect processes define the task and supply criteria for success and improvement at the task. The topographic map, or local database, contains the schema associated with the goal. The schema is both a program for performing interactions as initiated and maintained by the goal process, and the repository of all experiences at attempting the task. The variations and variation application process modify past experienced action attempts toward finding better ones.
3.3 Overview of the basic adaptive loop

What we will call the *base level*, or level 1, or the *basic adaptive loop*, of the architecture is depicted in Figure 3.3. The components in this figure are not the entire learning system, however, they are most of the components associated with the adaptive construction of a *topographic map* pertaining to one *goal*. The missing components, also associated with this topographic map, include another goal, corresponding to the process of functional assimilation, and another level of structure, which is similar to this level, called the *variation level*, and which will be discussed in chapter 4. We will not address how a system of multiple goals manages the coordination of these goals. For a recent treatment of the problem of goal coordination in artificial creatures, see (Maes, 1990), (Maes and Brooks, 1990), (Maes, 1991).

Referring back to Figure 3.3, we will describe each function block as it might become activated during the execution of one practise movement.

The entry point for the activation of this level of the system is the goal process, shown slightly above the center of the figure. The notion of goal that we are adopting defines the goal as a process internal to the organism. Recall our earlier definition of goals as those processes/mechanisms\(^{19}\) that relate action to biological needs. The analog of a biological need in this context is the need to make a successful reaching movement.

In the present case, a higher level process determines that this goal may be selected and enabled. However, the goal process must consider whether conditions are suitable for action. In this case, an object must be located. If an object is available, the goal considers the object's location and movement and passes these to the local database (the topographic map equivalent) as indicies for which action to take first.

The database will either retrieve the requested action (or action sequence) or interpolate an action, if necessary. Often, the database will retrieve what we call a schema instance, which in this simplified context is a sequence of actions rather than a single action. We will refer to a single action as a *control structure element* or CSE. This schema instance may be the "best action sequence taken so far" to accomplish the task under the given sensory conditions.

However, the system will usually not execute the sequence of actions, or schema, as it is. Instead, a modification will be made, which will usually be a relatively small variation of the retrieved schema. The system maintains a collection of variation operators for this purpose. These operators are very similar to the variation operators that are currently being explored in so-called genetic algorithms (Goldberg, 1989; Holland, 1992).

The modified schema is then presented to the motor apparatus for execution. This produces some physical action, such as a reaching movement. Such a movement may take several iterations around the basic loop, depending on how many CSE's there are in the schema instance (after modification). In any event, the goal process will keep track of

\(^{19}\) Attempts at understanding perception and action in the context of goals are becoming more frequent in the cognitive science literature. However, goals rarely get a formal treatment, and are usually identified as external to the organism and usually also identified as states rather than as processes. In our framework, there may be objects in the environment, with which the organism seeks to interact, but these should not be confused with the goals themselves.
whether the target has been hit, meaning that the movement can be terminated, or that the
target has not been hit, which can either mean that there are more CSE's left in the schema
instance to try, or more should be appended, or all of this is taking too long and the
movement should be aborted. Another subsystem on the diagram, the affect system,
assesses, in conjunction with the goal process, whether the action should be considered
successful, and what cost/payoff values should be assigned to the attempt.

The goal process then compares the assessed outcome of this action with its
unmodified precursor. If the action sequence is an improvement, the new action sequence
will replace the old action sequence in the local database. Otherwise, the changes that
produced the new action sequence may be stored along with the cost difference as an
inferior variation, of the old sequence.

It will come as no surprise that this description is somewhat simplified. For
example, control structures are not simply sequences of commands, but have considerably
more structure to include information about alternative or less successful subsequences.
As another example, there are generic kinds of cost functionals, in addition to the task-
specific ones mentioned so far. The difficulty with all cost functionals, however, is less
their computation than the attribution of change in the cost functional with specific actions
that may have occurred much earlier in time. The remaining sections of this chapter will
be concerned with describing this omitted detail including both the algorithms within the
function blocks, as well as the data structures (or representations) maintained and used by
the algorithms.

3.4 Control structure elements
3.4.1 Definition of control structure element

As we indicated previously, a control structure element (CSE) is the smallest unit
of conditional action in the system. A schema-instance, for example, consists of a
sequence, or more frequently a tree (a directed, acyclic graph), of control structure
elements. Each CSE, in turn, is a tuple of vectors of the form

\[ <\text{sensory-context, action, outcome, affect, goal, bookkeeping}> \] \[ .^{20} \]

The sensory-context vector contains the relevant sensory state components for this goal,
which for our tasks includes articulator position and velocity (2+2) dimensions, visual-
system target position and velocity (2+2) dimensions, and tactile position and contact
pressure (1+1 to 3+3) dimensions. The action-vector contains 4 dimensions for actuator
control (2 dimensions for either position or torque amplitude dimensions), 1 dimension for
onset-time and 1 for duration dimensions, and a dimension for controller type (discussed

\[ ^{20} \text{This is similar to Drescher's (1991) tuple structure for schemas. For clarity, we've denoted the possible layering of schema structure with different names for each layer of structure. In addition, we've added affect information, and a goal index to the structure as well as some bookkeeping information. Reasons for these extensions will be discussed later in this chapter. These differences are consistent with certain theoretical differences. For instance, we include a role for affect in our model, as well as a layer of structure for learning-to-learn. Another important difference is that learning in our system is goal-driven, by an assortment of goals processes which we are hypothesizing as genetically endowed (though we are not by any means assuming that all or even most goals are genetically endowed).} \]
below). The outcome vector has the same structure as a sensory-context vector\textsuperscript{21} and therefore the same dimensionality. The affect vector (typically 2 to 5 dimensions) contains measures related to learning progress. Lastly, there's a goal index (1 dimension), and some additional bookkeeping information.

3.4.2 Specification and interpretation of the contents of a CSE

Data in a component of the sensory-context vector of a CSE can be either a specific floating point number (within some bounds of that sensory device), or a pair of floating point values, which are interpreted as specifying a closed interval of values. Thus the sensory-vector can either specify an individual point in sensory-space, or it can specify a closed hypercube in sensory-space. The same, of course, is true of the outcome vector. However, the action vector is restricted to individual values, as are the remaining vectors, though this need not be the case for the affect vector. Allowing ranges of values in the context-vector provides a means of dealing with sensory noise and possible indirect effects of fluctuations in actuator performance.

The sensory-context is a representation of the conditions in which the action is known to evoke the outcome. In the case of one of the eye tasks, it says that if the eye-head mechanism is in position \textless head-pan-position-1, head-tilt-position-1\textgreater and if there is an object seen at \textless retina-x1, retina-y1\textgreater, then action \textless head-pan-actuator-to-pan-position-2, head-tilt-actuator-to-tilt-position-2\textgreater will result in the head being in position \textless head-pan-position-2, head-tilt-position-2\textgreater with the object at \textless retina-x2, retina-y2\textgreater. The CSE for this context-action-outcome tuple is

\[
\langle\langle\text{head-pan-position-1, head-tilt-position-1}, \text{retina-x1, retina-y1}\rangle, \langle\text{head-pan-actuator-to-pan-position-2, head-tilt-actuator-to-tilt-position-2}\rangle, \langle\text{final-position-position-error, final-velocity-error, movement-duration-as-cost}\rangle, \langle\text{goal-he2-stationary-target, bookkeeping}\rangle.\]

In this simplified example, the affect vector may indicate whether the object was brought within some reasonable distance of the fovea, and whether this was accomplished quickly or smoothly, etc.

Notice that there is no explicit representation of states of the system, or of the world. This is in keeping with our epistemological view that knowledge is the capacity to engage in successful interactions. The information in CSEs, to the extent that it captures state information indirectly, captures those states that have been found relevant by the system toward achieving its ends. However, the information in CSEs is meant to be in terms of what the system saw, what it felt, what it did, and what it then saw and felt.

\textsuperscript{21} In fact, it can be either a sensory-context vector, or a cached pointer to another CSE, whose sensory-context is appropriate, and which is the next CSE in the same schema instance.
3.4.3 Types of actions in a CSE

Control structure elements are implemented as objects. This means that they can contain the usual vector of parameters, but they can also contain procedures that access the particular vector component values stored within the individual CSE object. For example, each CSE has an "execute" function or method associated with it. This execute method reads the action vector within the CSE and translates the action values into some action function over the specified time interval.

One type of execute method will interpret the action vector as a vector of torques to be produced by the actuators over the specified time interval. Another type of execute method could interpret the action vector differently, for example, treating it as an equilibrium position vector. In such a case, the execute method implements a feedback controller that applies torques to correct the position error of the links over the specified time interval. Depending on the choice of execute method, different types of actions can be obtained. These include control for torque vectors (articulator torques for each link to be applied within some time interval), control for equilibrium position vectors (in joint-angle space), a combination of equilibrium position and velocity vectors (in joint space), or even the specifications of a feedback controller plus a vector of control parameters to be executed in a specific time interval. In different experiments we have used each of these possible actions. In one case, it was found useful to allow the system to pick the type of action (e.g. use several types of controller successively in one movement gesture) as well as the value of any action (actuator) parameters.

3.4.4 CSEs as data

The structural hierarchy of the system has topographic maps composed primarily of schemas, and schemas composed primarily of trees of CSEs. This puts the CSEs at the lowest level of structure in the system. CSEs being the terminal objects of what is primarily a tree structure contain most of the data of the system. The data is in the parameter vectors that fill each of the CSE's fields. For example, the action vector of a CSE may contain parameters specifying a pan angle and a tilt angle to command the eye-head controller. The data does not actually contain information about the eye-head position at any moment during the controller's actions. In a sense, the data only contains a goal, handed to the controller to seek. Thus, any approximation method using an action vector as data is approximating a surface somewhat removed from the movement of any limbs or actuators of the system.

While there is no explicit discretization of any of the components of vectors of CSEs (other than the truncation limits of the computer), there is a discretization of the schema and topographic map, in that it consists of a finite collection of CSEs, some distance apart in sensation-articulation-temporal space. Clearly, there are a variety of surfaces that can be approximated, depending on which of the CSE's fields are used as the data of this approximation. This arrangement provides a flexible means by which different approximations and representations can be combined.
3.5 Goal processes
3.5.1 Basic operation

The goal process provides the execution environment for the basic adaptive loop. In effect, the goal process is a kind of interpreter that retrieves a schema-instance from the local database, and executes it as a kind of program that specifies a sequence of interactions. Schema execution involves one iteration around the basic adaptive loop for each control structure element of the retrieved schema (modulo variation induced changes in the schema). In addition to the schema as program, however, there are both generic and task-specific computations that are performed. Some of these computations establish the context in which the schema instance is executed, others regulate the execution by starting it or stopping it, and still others evaluate the execution when it is completed, but before the schema is returned to the database. These additional computations are associated with the topographic map, which is to say that they are associated with the task. In object-oriented programming terminology, the extra computations are methods of the topographic map object, some of which are inherited by all topographic maps (generic methods), and other methods have definitions that may be unique to the task. Indeed, even the goal process itself is a method of the topographic map. It is the execute method.22

For example, every goal process contains the following categories of methods. Note that there can be more than one method per category.

Start-execution-predicates: (any one must return true to start)
(method-1) determines whether a specified object of interaction is still available (e.g. this is a subgoal of a higher level goal);
(method-2) determines whether there is a suitable object of interaction, if one has not been specified, puts object and method of identification in the current context;
(method-3) if no object is available, determines whether there is a schema instance that does not need an object for functional assimilation (practise).

Continue-execution-predicates: (all must be true to continue)
(method-1) is the object of interaction still present, if necessary;
(method-2) no higher priority goal has interrupted this goal?

Stop-execution-predicates: (any one must be true to stop)
(method-1) has frustration affect for reached a threshold for terminating the task unsuccessfully;

22 A further point is that each CSE is an object in its own right, with an execute method. Thus, a CSE can contain a basic adaptive loop, with its own goal(s) and learning process relative to this more restricted goal. In this way, a hierarchy of goals could, in principle, be explored and learned by the system. Are use of this feature has been limited: equilibrium position control, which is a regulatory loop, but without another layer of learning.
(method-2) if one of the cost functionals provides the successful stop condition, has it reached threshold cost for terminating the task successfully?

During-execution-methods: (called once per timestep)
(method-1) compute frustration related affect;
(method-2) check for and tag sudden affect changes (e.g. surprise);
(method-3) incrementally update remaining affects (e.g. cost functionals).

After-execution-methods: (called after stop-execution-method)
(method-1) compute all cost functionals associated with task;
(method-2) compute any other task-related variables needed for credit assignment;
(method-3) perform any credit assignment operations associated with the task, put results into appropriate CSEs of schema instance.

As an example of how these methods work, consider method-1 in the category of "During-execution-methods." This method-1 computes an affect variable, namely frustration, which is a generic (i.e. built-in) affect mechanism. Thus, the method and the corresponding variable, are introduced into every goal or task's context. This variable, which is a cost, starts at zero and is incremented with every timestep (timesteps are much finer than iterations around the loop). Another method-1, this one under the "Stop-execution-predicates" category, monitors the frustration affect variable, and when the value exceeds a threshold, the predicate returns true. The effect of these two methods is to put a limit on how long the goal persists in pursuing action. When the system gets frustrated enough, it gives up the current attempt, to start over another time. Furthermore, and "After-execution-method" adjusts the frustration threshold depending upon the experiences the system has had in achieving this goal. If the system is consistently not terminating successfully, it increases the threshold, if the system is consistently terminating successfully, it reduces the threshold, toward but not lower than a typical values at successful termination.

With this modularity, the learning system needs only one generic goal process. However, associated with each of these phases of goal processing, the generic goal process calls task-specific methods, which are the predicates and cost functionals associated with making sure that there is an object to interact with, or providing a substitute target in articulator coordinates for functional assimilation, determining whether the task is progressing properly, or whether it has terminated early (by hitting the target), determining whether some novel adjustment to the schema is needed, or determining whether the task should be aborted (successfully or unsuccessfully).

As mentioned above, each of these task-specific methods are stored within the topographic map, the repository of all task-specific functions and data.23

23 There are 5 sets of task-specific methods: a set for reaching to still targets, another set for eye movements to still targets, a set for reaching to moving targets, one for eye movements to moving targets, and lastly a set for variation learning.
3.5.2 Section Summary

From activation to termination, the goal process (1) determines whether (and when) to initiate an interactive program (a schema) on the basis of internal and external conditions, (2) issues queries to the local database (or topographic map) to retrieve an interactive program, a schema-instance, for execution, (3) requests the application of variations to the schema-instance producing a temporary replacement schema-instance for execution, (4) sets up an execution environment for the replacement schema-instance, (5) executes the replacement schema-instance CSE by CSE, (6) during which it monitors the progress of this execution relative to affective measures (cost functionals) and some generic internal criteria (e.g. if things are going much more slowly than normal, give up and start over), (7) determines when the execution is completed (successfully or not), and (8) prepares the new and modified, or the old schema-instance for re-insertion into the local database, depending the outcome of the completed execution.

3.6 Affect mechanisms
3.6.1 Introduction

The insight or hypothesis that guides our explorations of the mechanisms of affect is that affects are physiological signals that tell us how we are progressing in relation to our goals. Simply put, hunger and satiety are negative and positive affects, respectively, associated with the goal of maintaining blood glucose levels. A distinction is to be made between the affect signal that we associate with the experience of affect, and the affect mechanism, which is responsible for determining when to provide an affect signal, how strong the signal should be, and whether it is positive or negative (for those affects that have two poles). A more abstract way of putting the definition of an affect mechanism is that it is an internal mechanism that is associated with a goal, and which interprets the outcome of actions that have been initiated and selected (i.e. motivated) on behalf of the goal. Though an affect mechanism is internal to an organism, it often must sample the outcome of an action through the senses. However, many times relevant outcomes may be determined quite indirectly by pain mechanisms, or metabolic changes, such as due to exertion or stress. Thus, a lot of internal mechanism may be behind "interpreting the outcome."

In the discussion of the previous sections, we have only treated one affect, which we are hypothesizing is crucial for learning, namely frustration. In this section, we will discuss learning mechanisms we are hypothesizing to be associated with the affects of surprise, pleasure, and pain. These are, perhaps, generic affects, though their mechanisms sometimes have task-dependent features. In addition, we hypothesize that there are task-specific affects that are associated with the tasks of visual and tactile grasp, which may not be prominent phenomenologically, but which we believe must be present in some form for learning to occur.

As noted earlier, frustration is an affect that is associated with an expected length of time that a task should take. We experience it when the goal has not been achieved after the usual amount of time spent, and it increases in value with time.\textsuperscript{24} We have associated only one strategy with frustration, which is to abort the current goal-related activity, mark

\textsuperscript{24} Phenomenologically, "increasing in value" means that the experience is more intense.
the schema instance as unsuccessful, possibly apply a credit assignment algorithm to help determine why the schema was unsuccessful, and store the results of this processing back into the database. There are other mechanisms associated with the affect of frustration, some of which will be discussed in the next chapter. However, it should be mentioned that mechanisms associated with affects are themselves subject to change and learning. Both task-specific and more general strategies for dealing with frustration can be much more sophisticated than the simple mechanism suggested here. Such mechanisms show great individual differences, which suggests that strategies associated with affects are indeed subject to learning and development. This is an unexplored but promising direction for further research.

The next affect for which we propose a hypothetical mechanism is the affect of surprise (Tomkins, 1962; 1980; 1991). In our formulation, surprise is triggered/experienced at places in the control structure where the sensory context of the current CSE does not match the input coming from the corresponding senses. That is, the expected context at this location within the schema does not match the experienced context. Thus, it is a potential site of learning, as we will discuss next.

3.6.2 Mechanisms associated with the affect of surprise

A schema embodies the memory of a past experience of trying to achieve a goal with certain actions. At each step, or CSE, of a schema the <context, action, outcome, affect> information can be regarded as engendering an expectation that the outcome will occur when the action is taken, given the current context. Thus, if a schema is being executed and an outcome does not occur, then an implicit expectation has been violated, and the affect of surprise (in a biological organism) would be experienced. Thus, the negative sense of surprise indicates that in some way the existing control structure is not effective, and that this part of the control structure needs to be tagged as needing alteration or further elaboration.25

Under normal circumstances, executing a schema-instance involves checking that the sensory context matches the current sensory input, then performing the action for the specified duration, and then making sure that the outcome matches the outcome that has previously resulted from the action. The affect vector may or may not be updated at CSE execution, depending on the affect mechanisms used in the simulation. There are circumstances for which this pattern is too simple, one of which occurs when the newly experienced outcome is not the outcome currently stored in the CSE, but a surprise outcome.

We will consider a few cases to see how the system may distinguish a surprising outcome, from one that is slightly noisy but otherwise acceptable, or from an outcome that should be different because the context did not match up very well.

First, we will dispense with two cases that do not require further analysis. The first case is when the new outcome (as a point in \( \mathbb{R}^O \), where \( o \) is the the dimension of a

---

25 Though we haven't included it, there is obviously an opposite sense of surprise when an unexpectedly goal-positive outcome occurs.
context vector) is outside the stored outcome (as a region in $\mathbb{R}^0$), but is within some Euclidean distance set as a noise threshold. We will regard such an outcome as not surprising, but acceptably similar to the stored outcome. The second we will not regard as surprising is when the action stored is not the action taken because the current CSE has been selected to be modified by a variation operator (such as a random mutation). In such a case it might be surprising if the outcome were similar to the stored outcome. The latter case will be considered shortly.

The remaining cases, however, may be consequences of unexpected events in the external world, or internal to the organism (e.g. growth, sticky joints, etc.). We will consider those cases where the unexpected event is known to be due to factors other than noise. In such cases, there are several categories of deviations from the usual interaction pattern (1) the remembered sensory-context or outcome for a CSE (or CSEs) may not match the currently experienced sensory conditions or outcome, even though no variation has occurred so far in the schema, (2) a remembered sensory-context or outcome may not match experienced conditions but most outcomes including the final outcome and costs are not far off (i.e. all deviations are relatively small), (3) a remembered sensory-context or outcome may match the experienced conditions after the varied action, where a change would have been expected, (4) the target may be hit unexpectedly early in the CSE sequence, (5) the target may be missed altogether, when it was expected to be contacted.

Such deviations are dealt with as follows. First of all, the variations generally do not modify all the CSEs of a schema. Usually, they are intended to be variations of small amplitude, giving just enough change to determine whether the direction of change is correct prior to further search. With regard to pattern (1) when no variation has occurred, but a sensory-context or outcome is farther off from the remembered context than noise would suggest, then the CSE whose action just precedes the failed expectation is flagged for further alterations (search in action space) and learning. That is, on future runs of this schema-instance, the site of the expectation failure is the primary site of search to restore or improve both the outcome of the CSE and the final outcome of the schema-instance.

With regard to pattern (2), if the sensory-context or outcome had been off from past runs, but if the final outcome of the schema-instance was within a predetermined tolerance of the past final outcome, then the expectation failure is not regarded as significantly beyond system tolerance, and instead of further search, the sensory-context of the appropriate CSE(s) is (are) widened to include both past and present contexts.

In pattern (3), a change of outcome, that should have been produced by a variation, did not occur. This, too, is a form of expectation failure, suggesting that the magnitude of a variation was too small (e.g. the scale of search was too small). In such a case, further search at the site of the expectation failure is warranted to find the appropriate search radius. A widened sensory context may also be appropriate here.

---

26 Admittedly, this is a trivial way of dealing with noise, but it is minimally adequate for our focus on other issues. We will make the simplifying assumption that we know what the levels of sensory noise and actuator noise are. Consequently, our discussion will be limited to other internal or external sources of change, such as a growing arm or growing head-eye apparatus, a change of optics, etc.

27 The search scale associated with a CSE within a schema-instance is part of the bookkeeping information alluded to earlier.
A similar expectation failure can occur when the search scale at its smallest is still so large that small changes of outcome cannot be generated. In such cases, the system may actually become bounded away from the local cost minimum. Obviously, this situation is detected, the remedy is to reduce the search scale and continue practising.

Patterns (4) and (5) involve sudden changes in final outcome, where the target is not encountered when it would normally be (this is not necessarily a bad thing, it depends on the cost functionals associated with the task). Whether either of these cases should be regarded as an expectation failure depends on the nature of the variation that produced the outcome. However, if a very small change produced such a drastic change in final outcome, then more search, and perhaps a finer search scale is warranted at (or before) the flagged CSE.

These phenomena occur because our cost functionals can be discontinuous. For example, a very small change in trajectory of the arm can move it from hitting the target to missing it altogether. When missed, the tactile contact cost changes discontinuously.

In addition to using sensory-based expectation failure to guide temporal credit assignment, it is also possible to use affect-based expectation failures. In other words, analysis of expectation failures can also be applied to those values of the cost functionals which are computed incrementally with respect to CSE application. For example, if the accumulated jerk functional makes a sudden jump that is different from the level of jerk normally expected from one of the better movements, then this location in the movement can be flagged as a site of future search and elaboration of the control structure.

### 3.6.3 Presumed affects associated with cost functionals

**Assumptions concerning affects for the visual and tactile tasks**

It was assumed, from the beginning, that some form of pleasure was associated with a successful reach and grasp, as well as with a successful visual grasp. It seemed natural to taper the affect off gradually, as the location of target contact moved away from either the center of the hand or the center of the visual field, depending on the task. Thus, the affect mechanisms corresponding to pleasure affects for both task categories seemed to be naturally modelled by the cost functionals associated with position accuracy relative to the palm of the hand and the fovea, respectively (see sections 3.2.2 and 3.2.4, respectively). These cost functionals return no-value corresponding to no pleasure for contact sufficiently far from the hand or the fovea, respectively.\(^{29} \text{30}\)

---

\(^{28}\) This is equivalent to looking at the derivative of input-output (sensation-action-outcome) function of each CSE to see whether it's too big or too small.

\(^{29}\) The cost functionals that our learning algorithm can use do not need to be continuous, or even defined for the entire state space, which we do not explicitly discretize. The functionals only need to be continuous and differentiable for an open neighborhood around the optimal value, which must, of course, be a minimum (if viewed as a cost) within this neighborhood. There may be other local minima as well.

\(^{30}\) The cost functionals that can be used for the visual and tactile grasp tasks, are errors or measures associated with starting and ending locations (in either visual system or articulator coordinates), starting and ending velocity, energy expended, time taken, the jerk of the movement, etc. For arm movements, we often use ending position error (in tactile coords.), beginning and ending velocities (= 0), jerk for the
In the reaching task, there is also a functional associated with pain if the hand hits the target too hard. We used the final speed at contact (see expression 3.1.2) as the corresponding affect mechanism. The functional could just as easily been the force of contact. There is not corresponding pain-related mechanism for the visual grasp task.

Also for the reaching task, we have chosen an accumulated jerk functional because it has been demonstrated to characterize adult arm movements (Flash and Hogan, 1985). The suggestion to use the accumulated jerk functional to constrain an approximation process was made by Poggio (1990). Further, the functional could be thought of as providing a pleasure/pain signal associated with movement smoothness/jerkiness.

For the visual grasp task, we used a minimum-time constraint rather than a minimum-jerk constraint, for obvious reasons. However, if the visual task were a smooth pursuit task, then the minimum-time constraint would be inappropriate, unless it referred to the time it takes to capture the target initially.

These cost functionals were treated as constraints on the variation/approximation problem to be solved. Since explicit intermediate position and velocity errors were not assumed to be available, the burden of finding successively better solutions to the approximation problem was placed on the temporal credit assignment methods, approximation from the data so-far, and new methods of producing "reasonable" variations on existing solutions. To make the problem harder still, the choice in the arm tasks of using the tactile location of contact with the object on the hand or arm, rather than a visual measure, produced a cost functional that is not even continuous, let alone differentiable. Thus, the method of search (see Chapter 4) needed to be unusually robust.

As a final challenge to the system, a popular method of temporal credit assignment, known as temporal difference (TD) methods (Barto, Sutton, Watkins, 1990; Sutton, 1991; Sutton, 1990; Sutton, 1984), failed to work with the minimum-jerk functional. The reasons for this failure, as well as a possible remedy for the conflict, will be discussed in chapter 5.

Objections to the proposed relationship between affect and cost functionals

While surprise and frustration are familiar experiences, it is not obvious why we should refer to our task-dependent cost functionals (see section 3.2) as affects. The first objection might be that they are task dependent and that the definition of affect should only be applied to more general purpose experiences. The easy reply to such an objection is that hunger and thirst are affects, which are clearly task dependent. In fact their task specificity is what makes them useful.

A second objection is that it is not clear what physiological processes or phenomenological categories could be associated with such functionals.

To this objection we have two replies. The first is that for infants, there appears to be a distinct pleasure associated with tactile grasping, as well as viewing novel events. There is also a distinct pain with hitting a reached-for object too hard. We are indeed whole movement, and occasionally other terms for comparison purposes (e.g. maximum velocity, or an energy measure). For eye movements different constraints are appropriate. We use beginning and ending positions (in visual coordinates), beginning and ending velocities (= 0), and time for the whole movement (want minimum time, not minimum jerk).
assuming that the experiences of pleasure and pain are associated with goal-relevant functionals. Whether the specific functionals resemble ours is less important than the theoretical point concerning the role affect has for learning, which may be more clear for other goals, such as hunger and thirst. In any case, the research paradigm of exploring the relation between affect and learning has not been systematically addressed in any subdiscipline of psychology. Consequently, we can only acknowledge that there is much work to be done.

The second reply regarding the lack of obvious physiological manifestations or phenomenological categories for our cost functionals is that computations that fit our definition of affect may be performed entirely unconsciously and may require physiological or psychophysical experiments for verification.

**Comparison with other approaches**

Our approach of putting the affect mechanisms clearly inside the organism with an interpretive function, as well as making the organism’s behavior internally driven, are both in contrast to the reinforcement learning paradigm that the proponents of temporal difference methods associate with their work. The authors of this work regard a "reward" as being provided by the environment. In other words, they are putting both the goal function and action interpretation into the environment, as opposed to putting these functions into the organism. We feel that this is a gross error from the point of view of theoretical psychology. It is an unfortunate vestige of the once dominant S/R approach to psychology.

In fact, there is nothing inherent in the mathematics of temporal difference methods that prefers the empiricist framework that Barto, Sutton, and especially Watkins have assigned to it. Thus, our use of TD methods does not conflict with our constructivist approach to psychological theory.

As a final comment, affect has not received much attention in any branch of psychology, including, curiously enough, clinical psychology. Fortunately, this has begun to change (e.g. see Demos, 1982; 1986; Tomkins, 1962; 1980; 1991). What may be unique among our hypotheses about affect mechanisms, is the claim that affect mechanisms are essential to learning, certainly active, unsupervised learning. In fact, if our definition of affect is correct, it is not possible to formulate a viable theory of human learning without including affect mechanisms.

**3.6.4 Section summary**

Thus far we have introduced two task-independent affect mechanisms: one associated with expectation-failures, which we identified with surprise, and another associated with breaking off task pursuit when the attempt takes too long, which we identified with mild frustration. In either case, these mechanisms inform the system in a general way, about how the interaction is progressing in relation to its goals. Furthermore, both of these mechanisms are presumed to be built-in, general purpose mechanisms for bootstrapping a learning system. The psychologically equivalent affects are surprise and mild frustration, respectively. Additional task-independent affect mechanisms will be introduced in the next chapter on variation learning.
In addition, we modelled the cost functionals for both the saccade task and reaching task on presumed pleasure and pain mechanisms associated with each task. For the vision task contact near the fovea is presumed to be associated with the positive affect of pleasure. Similarly, for the reaching task, contact between hand and target is considered to be pleasurable if it occurs near the palm of the hand, or painful if the collision between hand and target is too hard.

Finally, the mechanisms associated with expectation-failure perform a kind of temporal credit assignment where the affects of surprise and frustration guide the assignment of credit. It is part of our theory of affect that the primary purpose of affect is to provide an internal mechanism for doing temporal credit assignment. Some of the mechanisms we have outlined could be part of a biological system's genetic endowment. If this is true, then the more general-purpose affect mechanisms could help bootstrap the learning system. Other methods of doing temporal credit assignment, though not psychologically motivated, will be discussed in the next section, and the next chapter.

3.7 Schema structures and the topographic map

Up to this point we have referred to a collection of schema-instances for the performance of a task interchangeably as a schema, a topographic map, and a local database. We have used the term topographic map to suggest that the data structures that we are implementing have both structural (i.e. organizational) and process similarities to topographic maps. To put this another way, though the learning algorithm is the same for each task, each task will populate its own local database of schema-instances. Recall that a task is specified by its goal process, its affect/cost functionals, its sensory apparatus, and motor apparatus. A task also has its own, initially empty, local database or topographic map to populate as experience with the task accumulates. Needless to say, our database analog of a topographic map has considerable structure to it. We will describe the crucial elements of this structure below. The description will be a simplification, not a full reflection the complexity of the implementation.\footnote{The implementation is entirely relational, but the relational structure is implemented with an assortment of tables, using variable length records, and several indexing schemes. This structure is somewhat less perspicuous than the point of view of a user of the database resource, as presented in this section.}

3.7.1 Schema-Instance Indexing

As mentioned earlier, a schema-instance is a finite sequence of CSEs that, when executed, constitute a movement gesture. Actually, there is more to the schema-instance structure than this, but we will get to that shortly. Schema-instances are indexed by several fields of the first CSE in the sequence. In all of our example cases, the relevant index fields are the initial position and velocity of the articulator in question (4 dimensions for the head-eye; 4 dimensions for the arm), and the initial position and velocity of the target in visual system coordinates (also 4 dimensions for the head-eye tasks; 4 dimensions for the arm tasks) and the goal (1 dimension). For fast access to the initial CSE by content, a 2d-tree is maintained using tuples with component values for each of the index fields. Recall that each CSE has an outcome vector, which specifies the context of the next
CSE in the sequence. Thus, retrieving the first CSE via the 2\textsuperscript{d}-tree index is efficient, and retrieving the rest of the CSE sequence, and associated schema-instance data, is also efficient.

Initially, an 8-dimensional (9 index fields - 1 goal field) open ball of small, fixed radius is used as a region of equivalent index (see Figure 3.4). If a target location, for instance, falls within this 8-ball for some existing schema-instance, then that schema-instance will be retrieved. This prevents the system from trying to push schema-instances too close together, though the context of a schema-instance can eventually widen this region.

3.7.2 Storage of best schema-Instance so far

What we have been referring to as the CSE sequence of a schema-instance is actually one of many sequences that have been executed on behalf of the schema-instance. However, we will designate as the \textit{trunk CSE sequence}, that part of the schema-instance data-structure that corresponds to the best CSE sequence found so far. That is, the \textit{trunk sequence} of the schema-instance has produced the best movement that the learning system has been able to find for the schema-instance's initial conditions, costs, and target location (see figures 3.4 and 3.6). However, this does not necessarily imply that the trunk sequence is the best movement that can be found, nor does it imply that this sequence will be executed without modification. It will always get some modification(s), though this modification may be quite small.

However, the trunk sequence of a schema-instance always has the distinction that modifications made as part of the search process for finding a better trunk sequence are always made to it. Thus, the trunk sequence is the center of any search neighborhood.\textsuperscript{32}

3.7.3 Storage of nearby "state" as experienced in movement attempts

As the term trunk sequence implies, the structure of CSEs within a schema-instance is a tree, not simply a sequence (see Figure 3.6). The reason for the extra CSEs is to retain some search information about movement attempts that have been made since the trunk sequence was established, but which have not produced further improvement. For example, Figure 3.6 depicts a sequence of CSEs that constitute a trunk sequence of the schema-instance. These are the CSEs connected by the horizontally running line, that indicates advancing time from left to right. Suppose that the search and variation mechanism has picked two positions in the sequence where it will vary the action values. These positions are marked A and B. At each of these positions, there are branches protruding from the trunk. Each branch corresponds to one CSE, which was a modification of the trunk CSE to which it is connected. Stored along with the modified CSE is additional information including the resulting values of the cost functionals at this attempt (possibly after credit assignment) and information about which other CSEs were also modified in this attempt. The collection of the modified CSEs in a single movement attempt can thus be treated as a unit by other computations.

\textsuperscript{32} Initially, only one trunk sequence was stored in a schema-instance at a time. We have since found that saving the n best CSE sequences, for some fixed n, as genetic algorithms do, often provides a more efficient search. In this case, we are saving an assortment of trees.
Retrieval:

Each schema has an index within a kd-tree using target location and sensory state as keys.

A query will return a schema in the DB if the keys match the query within a small radius.

Otherwise, a new schema is interpolated.

A schema contains (1) the best CSE sequence found so far for the movement, (2) a partial record of inferior attempts, (3) costs for the best and the variations, (4) the current state of the minimization algorithm on this schema, (5) a statistical history of variation ops. and their outcomes.

Figure 3.4: Also referred to as the local database, the topographic map is a collection of schema instances constituting a schema. A schema is a program for interaction with the environment, which operates in the service of a goal of the system. See Chapter 3 for a detailed discussion of the local database.
Figure 3.6: A graphic depiction of the tree of CSEs that are stored in a schema-instance. The upper figure shows the tree structure itself. The lower figure depicts part of a cost surface approximated using some of the CSEs in the tree structure as data points. The CSE's labelled A and B are the sites selected for modification by the current variation. Most commonly, this variation is a mutation operation, which means that the action vectors of each of the two selected CSE's are modulated with random action vectors. The small t's in some of the CSE's indicate that these CSE's are currently part of the "trunk" of the schema-instance.
The information in these branches of the schema-instance can be rather transitory. Once a movement attempt results in a better movement than the current trunk sequence, the movement's CSEs replace the trunk sequence in the local database for its particular initial conditions, target and goal. This replacement will make some of the branch information invalid. This can happen because in many cases the assigned cost information of the old branches depended on the combined outcome of the old branches and the old, unmodified parts of the trunk.

However, if the recent changes that improved the schema-instance accomplished the improvement by modifying parts of the trunk that were held fixed for the earlier branch explorations, such as at positions A' and B' in the figure, then the outcome of mixing those old branches with the new trunk is unknown. Consequently, the assigned credit for the old branches is unknown with respect to the new trunk, and is therefore deleted.

This turnover of state information may appear inefficient and wasteful, but in practise, it tends to work quite well. The reasons will become clear in the next few sections, but are partly due to the fact that the turnover of data in one schema-instance in comparison to the amount of data stored in the entire topographic map is relatively small, and that when the system finds a direction of modification that produces improvement, it can exploit the direction in several ways, such as searching via gradient descent along the vector of improvement.

3.7.4 Local approximation of the cost surface

The data in the tree of CSEs of a schema-instance can be used to discover and exploit structure inherent in the task that is being learned. In this section we will discuss two very simple ways of doing a kind of temporal credit assignment (TCA) using this branched data. Once a TCA computation has been selected, it can then be followed by a local approximation of the cost surface. A minimization procedure can, in turn, be run on the approximate cost surface. The minimum found on the cost surface will correspond to a CSE, or vector of CSE's, with lower cost than the respective CSEs in the trunk sequence. Figure 3.6 contains a simplified graphic depiction of this process.

To return to the simple TCA algorithms, the first method of doing temporal credit assignment is as follows. Suppose we have a trunk sequence of CSE's for schema-instance, S₁, designated CSE₁, ..., CSEₙ. For the sake of simplicity, assume that rather than the usual vector of costs associated with a task, we have a single cost dimension. Let the cost of executing CSE₁, ..., CSEₙ be c. Suppose that some subsequence is selected for repeated stochastic modification (i.e. mutation operations). Let this subsequence of n < N modifiable CSE's be

\[ CSE_{i₁}, ..., CSE_{iₙ} \]

Suppose

\[ CSE_{i₁}^l, ..., CSE_{iₙ}^l \]

is a modification of the trunk sequence, CSE₁, ..., CSEₙ, with the cost, c₁. This means that
\[ CSE_i^t = CSE_i, \quad \forall i \in \{k_1, \ldots, k_n\} \]

It also means that
\[ CSE_i^t = CSE_i + \bar{r}_i, \quad \forall i \in \{k_1, \ldots, k_n\} \]

where \( \bar{r}_i \) is a vector whose action components are non-zero and randomly chosen from some open hypersphere about the origin. All other components of \( \bar{r}_i \) are zero. Ideally, we would like to be able to take the cost \( c^i \) and divide it up among the \( \bar{r}_i \) so that we can get some sense of which CSE's and which action vector directions are likely to lead to lower costs for the schema-instance. Temporal difference methods are one category of technique for such problems. As we mentioned in an earlier section, temporal difference methods are not applicable with some of our cost functionals. Since it is not otherwise obvious how to do the credit assignment, we can combine all the modified CSEs that correspond to the same movement as one large tuple of data, and assign as a cost difference to this tuple, the difference between the cost of the trunk-sequence and the modified trunk sequence. Since we do not want to use the full dimensionality of a CSE, when it is not necessary to do so, we can select the context and action dimensions of each CSE to put into our combination tuple. If there are \( k \) context dimensions and \( m \) action dimensions, then the desired projection of a CSE can be denoted

\[ \text{project}(CSE_i^t) =< c_1, \ldots, c_k, a_1, \ldots, a_m >^j = < (c_1)^j, \ldots, (c_k)^j, (a_1)^j, \ldots, (a_m)^j > \]

where we have put the CSE's super and sub-scripts on the tuple itself as a simplified notation. The notation means "project the CSE of the \( i \)th position in the \( j \)th modification of the schema-instance to a tuple containing its context and action components."

The projected tuple components can then be combined for all the modified CSE's of the schema instance. The composition of the combine and project operations is denoted,

\[ \text{combine\_project}_{1..n}(CSE_{1}^t, \ldots, CSE_{n}^t) = \]

\[ \text{combine}_{1..n}(\text{project}(CSE_1^t), \ldots, \text{project}(CSE_n^t)) = \]

\[ < (c_1)^{x_1}, \ldots, (c_k)^{x_1}, (a_1)^{x_1}, \ldots, (a_m)^{x_1}, \ldots, (c_1)^{x_n}, \ldots, (c_k)^{x_n}, (a_1)^{x_n}, \ldots, (a_m)^{x_n} > \]

For example, if two CSEs were modified, and if there are 6 context dimensions, and 2 action dimensions per CSE, then the combined and projected tuple would have 16 dimensions.

If \( J \) modifications to the trunk sequence consisted of modifying the same \( n \) CSE positions, then the \( J \) data samples of the form

\[ \text{combine\_project}_{1..n}(CSE_{1}^t, \ldots, CSE_{n}^t) \text{ with cost } c_j, \]

can be treated as example data defining a cost surface pertaining to the modifications of this schema instance. The unmodified trunk sequence is also included as data for the cost
surface. These tuples can have a very high dimensionality, especially if a large number of CSE's per schema are modified. In extreme cases, where the dimensionality must be reduced, eliminating some or all of the context dimensions can still produce useful results. In the most extreme case, only the action dimensions and costs contribute to the cost surface.

To approximate the cost surface, we have used the GRBF method presented earlier in this chapter, where each data tuple above is the center of a gaussian radial basis function.

Figure 3.6 depicts a schema-instance and some of its modifications in the upper portion of the figure. Below it, a small piece of the approximated cost surface, corresponding to the modification of one CSE, is shown. Minimization on this surface would then produce 1 new candidate CSE for this modifiable position in the schema-instance. Executing the schema-instance with the minimized CSE in the modified position, should produce a lower cost outcome than the unmodified CSE sequence.

On the other hand, when there are J modifications to the trunk sequence, as in our computations above, then the tuple produced by minimization on the cost surface will contain the <context,action> information for J new candidate CSE's embedded in the single tuple. These J CSE's would be inserted into the J modification positions of the trunk CSE sequence of the schema-instance. When this schema instance is executed, the resulting cost should be lower than any cost achieved so far.

The new cost may not be lower, and when it is not the cause is most often the choice of sigma(s) for the gaussians in the RBF network. Where the approximated cost surface has a minimum is highly dependent on the assumed smoothness of the surface, and the smoothness constraint is implicitly specified by the choice of sigma(s). We will discuss methods we have used for picking sigmas at the end of this section.

Another way to find a cost surface from the same data would be to consider each CSE modification of one movement attempt as contributing a different amount to the cost difference between the modified sequence and the trunk sequence. The amount of cost difference assigned to a CSE would be in proportion to the CSE's euclidean distance from the trunk CSE it modifies. Thus, credit for the final cost is assigned to CSE's according to their distance from the trunk. While this method is quite crude, it does keep the dimensionality of the local cost surface approximation down to the dimension of one CSE, rather than the combined dimension of many CSEs, since a separate cost surface can then be constructed for each modification position.

Whichever method produces data for the cost surface(s), the subsequent steps are the same. Each dataset corresponds to the cost surface in the vicinity of CSEs that are being experimentally modified. All of these CSE-related tuples (not restricted to those CSEs of the best movement) are used as centers of gaussian RBFs. The system then tries to do a limited extrapolation of potentially lower cost CSEs by performing a minimization (via gradient descent) on this cost surface. In the first method of doing credit assignment, this minimization will yield one high dimensional tuple that will specify several CSEs modifications. The system would then try this modification of the trunk sequence to see whether it does in fact improve the movement by lowering total cost. Similarly, the second method of TCA would yield one CSE modification for each dataset. In other words, one CSE modification is found for each locally approximated cost surface. All or
some of these CSEs could then be used to form one or several modifications to the trunk sequence. In practice, the first method of combining CSEs into one large tuple worked extremely well for the arm movement task. The second method worked well for restricted cases, where only a small number of positions were modified (1 or 2), and modifications were reasonably close to the trunk.

In effect, performing a minimization on a locally approximated cost surface can substitute for a certain number of "external" minimizations through trial-and-error movements. In this way, the system can use approximation of the cost surface to substitute for actual movement trials with the hardware. These routines work adequately with fixed sigmas, but adaptively varying sigmas to keep the estimated CSEs consistent with the results of actual trials may improve the system's overall rate of convergence. We have not made many runs with adaptively modified sigmas as yet. The method we have implemented involves making each sigma proportional to the average distance to other data points. A large number of comparison runs would be needed, however, to compare its performance with our other methods.

3.7.5 GRBF Interpolation of new CSEs from existing data
Choice of data points for approximation

In the previous sections we considered cases where a schema-instance, with indicies that match the current target location and starting state, is already in the local database. However, it is frequently the case that the current target location and starting state do not match any existing schema-instance's index very closely. In such cases, a new schema-instance is interpolated (or extrapolated). First, we will consider the data that is available in the system for this interpolation.

Every CSE in the database, whether it is part of a trunk sequence or not, contains <context-action-outcome-affect> information, that has been "experienced" at least once. Since noise is not a major consideration, we could use all the CSE data available, depending upon what function(s) we wanted to interpolate. However, this approach would be prone to overfitting difficulties as well as a kind of noise in that most of the CSEs in the database correspond to misdirected attempts. As a result, we chose to use only CSE data from trunk sequences for our interpolations (see figure 3.4). Consequently, every <context-action-outcome-affect> datum, that contributes to an interpolation, is part of a best movement found so far. For reasons that will become clearer after the discussion of variation operators, this data will tend to be spread out relatively evenly in time, without tight clustering. As a bonus, this approach works extremely well for simulations where growth of the limb occurs. It does not add to the dimensionality the problem by including limb mass and size dimensions explicitly. On the other hand, it does allow the GRBF centers to move, as the system adapts to the changes produced by growth.

Moving centers

One characteristic of our algorithm for generating sequences of CSE's is that it generates many more data points than there are centers or knots used in an approximation.

---

33 Even with the head-eye hardware, the noise is quite low for the vision hardware (cameras and frame buffer) and low-to-moderate for the head-movement system.
Those centers that have any persistence tend to move about quite a bit, and in general, the number of knots increases with time. This increase continues up to a saturation point, beyond which it becomes very unlikely to introduce new knots (see discussion in subsequent subsections). However, the usual overfitting difficulties encountered with RBF approaches, do not occur with our approach. Overfitting can occur when the number of data points becomes large compared with the number of centers actually needed by a basis that spans the solution space. The centers or knots used in an approximation in our system are only those centers that correspond to the best-schema-so-far with respect to a particular goal. These centers tend to be reasonably evenly spaced within a schema (though this can vary depending on the problem) as variation operations dynamically alter the temporal or spatial mesh. Further heuristics that determine when to interpolate in regions of schema space between schemas (see next subsections) put a limit on how densely schemas can be "packed" within the database. Data that would grossly overconstrain the approximation problem drifts out of those parts of schemas that are currently denoted as the trunk of the schema. Since all the remaining centers are also data points, there is no need for a reformulation of the approximation problem for #data-points >> #knots.

**Choice of interpolation strategy**

So far, we have chosen which subset of the data to use, but we have not discussed which dimensions to interpolate for. Clearly, we need to get a sequence of actions, at the very least. However, once we have interpolated the first action, which we can do because we have the starting state information, we will not know what the outcome of that action will be. We could interpolate the outcome as well, and use it as the next state (context) vector to interpolate another action, and so forth. However, here we have the problem of accumulated errors, which can be quite serious. Accumulated errors are most serious early in the learning process, or during conditions of growth.

**Choice of when to interpolate and when to save the interpolant**

With these considerations, we chose to do the interpolation on-the-fly as each intermediate state becomes available after each action is taken. To do this interpolation, CSEs of the existing schemas are used as data, where each CSE's <context> vector is a center of a GRBF. The interpolated sequence of CSEs may be stored in the database after it has been executed. Subsequent retrieval of this schema will trigger the usual learn-by-trial-and-error methods of improvement.

If every new CSE sequence that is interpolated were saved in the database, there would still be the possibility of overfitting, even though we are not including most of the CSE's that have been generated in the learning process. As a simple way of regulating the likelihood of overfitting, the database does not always save an interpolated CSE sequence when the interpolated sequence yields a similar performance to nearby CSE sequences. By similar performance, we mean having similar values for the cost functionals.

If the interpolated sequence is worse in cost than the average costs of several near neighbors, then it is assumed that the gaussian centers of the neighbors are too sparse and the interpolated sequence is stored in the database. To be stored in the database, a
schema-instance object is created to contain the CSE sequence, and a retrieval index range is assigned to the schema instance.

There is one problem with this algorithm. If overfitting has already begun to develop, then quite possibly an interpolated CSE sequence could have a high cost due to the overfitting. In that case, saving the interpolated CSE sequence would make the overfitting worse, and so on. A simple check for this occurrence is to look at the density of CSE's in regions of the database, and compare the density with the rate at which new CSE's are interpolated and saved. If the rate at which interpolated CSE's are saved does not fall off with increasing CSE density, then overfitting is likely to have occurred. Currently, the system returns an error when this happens.

**Frequency of computing the matrix, G**

A drawback of our approach is that every movement can potentially alter the schema, and potentially require a recomputation of the matrix, G (see equation 3.1.c). We have managed the problem heuristically. When the database is small and relatively sparse, we recompute G, but only after successful variation operations. In other words, G is not recomputed for each successful stochastic search (mutation) operation. Successful variation operations are significantly less frequent than successful mutations, which in turn are a small fraction of all mutations. As the database grows it can be segmented into subsections (possibly overlapping) to reduce the size of the effective G, when it does need to be recomputed. Beyond this, G's recomputation can be triggered both by the amount of change that has occurred in the database since the last computation, and partly stochastically. We have not pushed the system to the point where these heuristics have been taxed, so we can not assess their effectiveness.

**3.7.6 Balancing practise and interpolation**

One issue that our application of generalization by approximation raises is how all the trajectories in the system can be brought to about the same level of performance. A more difficult question is how all schema-instances can be brought closer to producing optimal trajectories, when some trajectories may get practised far more than others.

We have supplied a mechanism that attempts to solve this problem by mimicking its counterpart in human development. The corresponding process was observed and named by Piaget as functional assimilation (see Ginsburg and Opper, 1969). The mechanism that we have hypothesized as a model of functional assimilation involves a goal process that specifically directs self-initiated forms of practise when other goals of the system are not operating. There is a separate functional assimilation goal process for each task and associated topographic map, though the structures of these goal processes are nearly identical for each task.

We will briefly discuss several sub-problems with the problem of self-initiated practise. The first sub-problem is to find regions of the topographic map that need more experience. The second sub-problem is to find targets for the needed experience. The third sub-problem involves re-orienting the system so that an existing target is moved into a region of the topographic map that requires experience. The fourth sub-problem involves determining when a visible target is not actually necessary to exercise the topographic map, and selecting an internal substitute for the non-existent target.
The first problem of identifying regions of the database that require filling is relatively easy for both the visual and tactile grasp tasks. Each schema-instance in the database has a visual target location as part of its context data. The density of visual targets in the system's visual coordinate space can be used as a gross measure to identify regions of inexperience. In this case, inexperience corresponds to a region of visual space with low density of visual target indicies. This only works as an initial crude measure. When visual space is reasonably covered with targets, a second measure must be used. The second measure looks at the values of the cost functionals assigned to the schema-instances across visual space. Regions where the cost functionals are high require more practise.

Once regions of needed practise have been identified, the system needs to check to see whether there is a target(s) available in one of these regions, or whether anything that can be used as a target is available. This requires a sampling of the visual environment and a simple check to see whether one of the possible targets is in one of the identified regions needing experience. If there is a target in one of the regions needing practise, the current articulator position can be saved, then a practise movement toward the target can be made, and the system can then return itself to the saved starting position. If the functional assimilation goal is still active, another practise move will be made. On the other hand, if there were targets in the visual scene, but none happened to fall into a region needing practise, then, in the case of the visual task, the system can make a random movement to try a different visual perspective. The system can then check again to see whether there are any targets in regions needing experience, etc. In the case of the reaching task, the system can not change its visual location w.r.t. the target, but it can change the starting location of the hand to see whether there is a trajectory region that needs more experience. These comments address the second and third problems of finding targets to practise with.

For the fourth problem, it is possible with either the visual or tactile tasks to practise without a visible target in certain circumstances. These circumstances occur when there are movements that have been practised enough to minimize the final position error to a very low value, but where the other cost functionals could be minimized further. In such circumstances, a visual target position is not needed because the system has learned the articulator coordinates that correspond to the target location, if there were one. In other words, the system can practise movements using the final articulator position that it knows corresponds very well to the final target position, if there were a target. This works because our approach to movement learning does not specifically require visual guidance of the articulator during the gesture, only a visually located target. Consequently, practise for some movements does not even require the presence of a target, when it can use the now known beginning and ending articulator locations of the movement.

To summarize, the goal of functional assimilation seeks to lower the costs of the least practised schema-instances, and to widen the experience of the system to new regions of the topographic map, as targets become available. This model of functional assimilation cannot seek to give the system exhaustive experience in state-action space, but it can, to a limited extent, "fill in" unexplored regions of the topographic map. This
process is ultimately limited by target availability. Mostly, the functional assimilation goal will search the state-action space in the vicinity of trajectories it has needed to construct.\textsuperscript{34}

3.7.7 Summary of local database operation

To summarize, the database portion of the system can learn a large number of sequences of CSEs, called schema-instances. A schema-instance, when executed, will produce a very narrow range of trajectories, though it is probably better to think of a schema-instance as corresponding to a single trajectory. A collection of schema-instances corresponding to a topographic map are stored in a relational database.

Schema-instances are indexed by target-location and articulator-starting-state with a 2d-tree. When a target appears, the system looks for the nearest schema-instance that matches the target location and starting state. If there is a schema-instance in the database, and if it is a short distance away in state space, then that schema is retrieved and executed. Otherwise a new schema is interpolated on the fly as each intermediate state becomes available after each action is taken. If the interpolated schema-instance produces a trajectory that is as effective as its neighbor schema-instances, or if the database is sufficiently dense in this region of the topographic map, then the interpolated schema-instance is not stored in the database. This reduces the problem of oversampling. Otherwise, the interpolated schema-instance is stored as a new item in the topographic map.

There is a class of goal process whose purpose is to promote evenly filled topographic maps, as much as is possible. We have identified this class of goals as corresponding to the process of functional assimilation as first identified by Piaget.

3.8 Chapter summary

In this chapter we have described a simple architecture for an internally-driven, goal-directed, interactive learning system. To apply the system to a specific learning task, the system is given a set of senses, motors, affects implemented as constraints, and variation operators. The system then generates (inter)action sequences, randomly at first, to try to minimize its affect-related constraints. The system usually discovers an interaction sequence that will improve at least one of the constraints after some trial and error. Most trial-and-error attempts consist of small, random, parameter perturbations of its best interaction sequence so far. However, in time it will get stuck with this type of "mutation" operation. When this happens, a more drastic modification of the interaction sequence is needed. What we are calling variation operators are used for this more drastic change in the interaction program. Usually, the application of at least one of these operators, in a given context, will enable the system to resume progress in learning its task. When this happens, the altered interaction sequence is saved as the best current solution to the task. The system then proceeds to modify this new solution with mutation.

\textsuperscript{34} Notice that this problem does not occur in other approaches to sensorimotor learning, because other systems are specifically trained in such a way that they get a fixed set of optimal trajectories. Additional trajectories are often linearly interpolated. Consequently, the situation where a very nearly optimal trajectory is stored close (in state-action space) to a mediocre trajectory doesn't occur in these other systems.
operations. In time, it will get stuck again and additional variation operators will be needed to restructure the interaction sequence so that learning can resume.

In the next chapter, we will describe the set of variation operators that the system uses to solve interactive problems. In addition, we will explore ways in which the program implicitly specified by the sequence of effective variation operators can be learned and reused as a kind of practise strategy for the task.
Chapter 4  The Variation Level of the Interactive Architecture

4.0 Introduction
This chapter contains a description of the variation level of our unsupervised learning algorithm. This level is the source of all change to existing schemas. Such change is always accompanied by a trial-and-error application of the changed schema. The chapter is divided into 2 main topics. The first topic is a description of what we are calling variation operators, which are transformations that modify a schema in the search for a better schema for a specific task. Variation operators have a close resemblance to genetic operators. Indeed, the collection of operators we have found to work for our problems are a combination of operators we discovered by our own trial-and-error experiments with the system, and operators from the literature on genetic algorithms that extended our set toward greater generality. However, while our operators resemble genetic ones, our search algorithm is not, strictly speaking, a genetic one. We did not restrict our operators to finite coding from a finite alphabet. Instead, search is conducted in a finite dimensional hypercube that is not discretized. Secondly, we rely more on mutation operators than genetic algorithms usually do. That is, stochastic search plays a slightly greater role in our algorithm. However, our biggest departure from the genetic approach is the learning and generalization of a subsequence of the variation operators that were applied during the construction of schema instances. During our experiments, we found that certain subsequences of variation operators would promote quicker learning of reaching movements to targets in new parts of the articulator space. The improvement in learning was especially noticeable in regions of articulator space that were sufficiently distant from sites of previous learning that interpolation or extrapolation would not cover. Thus, the second topic of this chapter is an algorithm capable of learning variation sequences by trial-and-error and generalizing variation sequences to form what we will call a \textit{practise strategy}. This algorithm runs in conjunction with the basic adaptive loop as it is learning the task.

4.1 Search Operators and Variation Operators

4.1.1 Definition of search operators
Recall from the previous chapter that a schema instance is composed of a collection of control structure elements (CSEs). This collection is always organized into a tree structure. Execution of this tree structure begins at the root node and proceeds CSE by CSE to a leaf CSE of the tree. There is one path through the tree that represents the current minimum cost path. This path is referred to as the trunk CSE sequence of the schema instance. Let $S_i$ be the $i$th schema instance. Mathematically, $S_i$ is a tuple of the following form

$$S_i = < \text{CSE-tree}_i, \text{trunk-sequence}_i, \text{search-set}_i, \text{search-parameters}_i ... >$$

where CSE-tree$_i$ is the tree of CSEs, trunk-sequence$_i$ is an ordered list of identifiers (in this case unique integers which are database indicies) of those CSEs comprising the trunk
path through the CSE-tree, and search-set is a list of identifiers of those CSEs in the tree whose actions can be modified. The ellipsis indicates additional components to the tuple which will be introduced as needed for the discussion. Let T be a transformation from the vector space of schema instances to itself, and let \( \{ T \} \) be the collection of all such schema transformations. Suppose that \( T \) maps \( S_i \) to a schema instance \( S_i' \), and that \( S_i \neq S_i' \). We will separate the collection of schema transformations \( \{ T \} \) into two disjoint subcollections of transformations with the following definitions.

**Definition 4.1:** A search operator, denoted \( T_S \), is a member of \( \{ T \} \) such that when \( T_S: S_i \rightarrow S_i' \) then for some collection of CSEs, \( \{ \text{CSE}_j \} \), in trunk-sequence \( i \) of \( S_i \) and in search-sequence \( i_j \), if \( T_S: \text{CSE}_j \rightarrow \text{CSE}_j' \), then \( a \neq a' \), where \( a \) is the action vector of \( \text{CSE}_j \) and \( a' \) is the action vector of \( \text{CSE}_j' \). Furthermore, all other components of \( S_i \) and \( S_i' \) are equal.\(^1\)

In other words, a search operator alters a schema only by altering the action subvectors of CSEs in the current trunk-sequence and in the current search-set. No other changes are made to the schema or to its tree of CSE's. The idea is that the action components of the trunk sequence of CSEs span an action (vector) space. Search operators move around within a region of this space without changing the basis. The effect of these operators is to explore actions within a region of space that is implicitly defined by the control structure. The implicit specification of the search region is determined by the number of CSEs in a schema, the spatial mesh of the changes to action vectors, and the temporal mesh specified in the durations of the CSEs in the schema. Search operators are analogous to mutation operators in genetic algorithms.

### 4.1.2 Definition of variation operators

Our second class of operators we will refer to as variation operators.

**Definition 4.2:** A variation operator, denoted \( T_v \), is a member of \( \{ T \} \) that is not a search operator, and which performs one or more of the following changes.

1. If \( T_v: S_i \rightarrow S_i' \) where \( T_v: \text{CSE-tree}_i \rightarrow \text{CSE-tree}_i' \) then \( \text{CSE-tree}_i \neq \text{CSE-tree}_i' \).
2. If \( T_v: S_i \rightarrow S_i' \) where \( T_v: \text{search-set}_i \rightarrow \text{search-set}_i' \) then \( \text{search-set}_i \neq \text{search-set}_i' \).
3. If \( T_v: S_i \rightarrow S_i' \) where \( T_v: \text{search-parameters}_i \rightarrow \text{search-parameters}_i' \) then \( \text{search-parameters}_i \neq \text{search-parameters}_i' \).

Thus, variation operators will, in general, change the space spanned by the actions of the trunk-sequence, or at the very least, they will change the region of search. In other words, variation operators change the control structure in such a way that they change the action space being searched.

---

\(^1\) N.B. This definition should also say that search operators preserve the CSE-tree structure. Nor do search operators change the temporal duration of the action vector.
Let \( g(x) \) be a multivariate function on a region \( A \subseteq \mathbb{R}^n \). Let \( x \in A \) be an interior point of \( A \). Define the finite set of integers \( I_n = \{1,2,\ldots,n+1\} \) and let \( P = \{P_1,\ldots,P_k\} \) be a partition of \( I_n \). I.e. \( P \) is a disjoint collection of non-empty subsets of \( I_n \) whose union is \( I_n \). \( \forall P_i \) pick \( \omega_i \in \mathbb{R}^+ \), and let \( \Omega = \bigcup \omega_i \), which is called the noise list.

Pick some starting vector, \( x \), and let \( g_{\text{current}} := g(x) \).

\( \forall P_k \in P \)

Loop until \( \omega_i < \text{small\_threshold} \), \( \forall i \in I_k \)

Construct \( v \in \mathbb{R}^n \) stochastically by,

\[ v_j = \text{random}( -\omega_i, \omega_i ) \text{ if } j \in P_j, \text{ and } v_j = 0 \text{ otherwise.} \]

\( g_{\text{temp}} := g( x + v ) \)

if \( g_{\text{temp}} < g_{\text{current}} \) then {

\( g_{\text{current}} := g_{\text{temp}} \)

\( x := x + v \)

\( \omega_i := 2 \omega_i \)

do a 1-dimensional gradient descent to find

\( r \in \mathbb{R} \) such that \( g(x + rv) \) is minimal

\( g_{\text{current}} := g(x + rv) \)

} else

\( \omega_i := \omega_i / 2 \)

**Figure 4.1:** This is a combined search algorithm that uses the stochastic search technique of Caprile and Girosi to find better points than the current best point (searching for a minimum) \( g_{\text{current}} \). It searches in open hyperspheres around the current best point. When it finds a better point than the current minimum, then it searches along the vector from the previous best point to the new best point to find and even better point using gradient descent (Brent’s parabolic method). Whenever a new best point is found, the radius of search is widened by a factor of two. Whenever a random point within the current hypersphere is not an improvement over the current best point, the diameter of the hypersphere is halved. Thus, it widens the search in relation to the best point when the search has been successful, and narrows the search when it is unsuccessful. Removing the 1-dimensional gradient descent part of the search produces exactly the algorithm of Caprile and Girosi. When the search radius reaches some sufficiently small value, the algorithm is restarted at a radius that is just a little wider than the average radius at which it has had prior success. A variation operator can reset the search radii to their maximum values.
(1) Split nth CSE in time

(2) Combine adjacent CSEs into one

(3) Append, delete, or insert CSE(s) into schema

(4) Select CSE controller type (e.g. torque-vector, or equilibrium-position)

(5) Select pattern of CSEs to mutate

(a) 1 CSE at a time
(b) 2 adjacent CSE's
(c) cluster of n adjacent CSE's
(d) symmetric pair of CSE's
(e) symmetric cluster of 2n CSE's
(f) all CSE's in schema

(6) Alter maximum or minimum search parameters (e.g. limits)

(7) Select search algorithm (e.g. stochastic, gradient descent, combination)

(8) Crossover at Schema-A, position n, and Schema-B, position m

**Figure 4.2:** This is a sample of variation operators that can be applied to base level schemas. Genetic algorithms emphasize mutation operations less and tend to favor a crossover operator.
4.1.3 Examples of Search operators

As examples of these classes of operators, we have adapted several search algorithms to function as search operators in the system. One is a stochastic search algorithm of Caprile and Girosi (1990), which is described in Figure 4.1. A second search algorithm we have used is gradient descent. The third type of search is a local GRBF approximation of the cost surface, described in the previous chapter. Lastly, the most effective search algorithm for these problems that we have tried is Caprile and Girosi's stochastic search algorithm modified to do a 1-dimensional gradient descent along vectors that are discovered to have improved the minimum (see Figure 4.1). The reason for trying these alternatives is that by simultaneously trying to minimize several cost functionals, usually 3 for any given task, the system is searching a surface that has many local minima. Thus, gradient descent alone is going to "get stuck" time and again in these minima. However, the combination of stochastic search, gradient descent and the effects of the variation operators taken together appear to give the system some capacity to find its way out of local minima, as genetic algorithms do.2, 3

4.1.4 Examples of Variation operators

Some variation operators that we have tried and have found to be useful for the two grasp tasks appear in Figure 4.2. These include operators that modify the length of the schema by inserting, deleting, appending and truncating CSEs in the trunk of the schema (type 3 in the figure). Other operators modify the temporal mesh of the schema. These are types 1 and 2 in the figure. For example, the temporal mesh can be refined (locally) by splitting CSEs in half, by replacing one CSE of duration, d, into two CSEs each with duration d/2. The temporal mesh can be coarsened by combining two adjacent CSEs of duration d1 and d2 into one CSE of duration d1+d2. In each of these cases, it is not always obvious what should be done with the action vectors of the split or combined or inserted CSEs. Our approach has been to assign values to the action vectors that produce the least difference in the action function of the CSE sequence after the modification. For example, operators of type 1 may split a CSE in half; but the action vectors are unchanged, so the function computed by the CSE sequence remains the same. The type 2 operator does change the function computed by the CSE sequence in going

---

2 The best minimization algorithm tried so far is that of Caprile and Girosi (1990), modified to do a Brent method 1 dimensional gradient descent in the direction just found to do best so far. In the way of trying other minimization routines. In comparison with gradient descent, this enhanced stochastic algorithm tends to be slower, but it doesn't mind the discontinuous functionals, and it does eventually find its way out of local minima. It is not hard to see that gradient descent gets stuck in local minima, which our plentiful with our cost functionals. More interesting, would be to see how a genetic algorithm with crossover, for example, compares with the others. A genetic algorithm would fit particularly well into our learning architecture. In fact, the only difference between our basic adaptive loop with variations and a genetic algorithm is the addition of (1) a crossover operation, and (2) saving more than one "trunk sequence" in each schema instance. That is, rather than saving the best 1 movement found so far, we would need to save the N > 1 best movements.

3 We also tried a form of TD at this level of the system. TD contains a stochastic component, but as we indicated in the previous chapter, its minimum-time characteristics conflict with minimum-jerk, minimum-torque-change and other functionals that tend to minimize energy.
from two CSEs to one CSE whose action is the average of the two CSEs before combination. This change in function computed by the CSEs will obviously tend to be small as the temporal mesh gets smaller. Our intention is to minimize the change that the variation operator produces in the schema instance, so that subsequent mutation operations (stochastic search) will determine whether the modification is a good one as well as what magnitude and direction of change are best, within the new search space.\(^4\)

In addition to the variation operators discussed so far, we have also found it useful to supply operators that can (1) change the search strategy by switching among several search algorithms (operators of type 4 in the figure), (2) change the search pattern, which is to change which CSEs within a schema are marked for search (operators of type 5), and (3) change the search limits, that is, the upper and lower bounds of each dimension searched (operators of type 6).\(^5\) Finally, another set of variation operators is intended to establish the initial conditions or initial form of the CSE as the system begins to build a new schema instance. These operators include (4) selecting the initial CSE count (not in the figure), (5) selecting the initial CSE type, (6) selecting the sensory dimensionality of the CSEs (the default is to use all available dimensions). Another operator (7) that is a variant of the last operator selects which cost functionals to attend to by altering a weight vector, \(w_c\), which is used in computing a single cost from the vector of costs, \(c = <c_1, c_2, \ldots, c_n>\), one component for each cost functional associated with the task. Of course, the single cost is obtained by taking the inner product, \(w_c \cdot c\). Minimization is done with respect to this single cost scalar.

One class of variation operators, those that set the search pattern (type 5 in the figure) may provide some insight into the nature of "movement units" found in the infant reaching literature. If, for example, a search pattern of \(n=20\) temporally adjacent CSEs is selected for mutation, while the other CSEs are held fixed. In addition, if the search cluster is changed to a different, non-overlapping \(n\) CSEs every once in a while. Furthermore, if these clusters form a minimal cover of the schema, then we have partitioned the CSEs of the schema into movement units. Consequently, the dynamics of the reaching movement during learning should resemble that of infant reaching dynamics. Since jerk will be minimized within a movement unit, while the other movement units are held fixed, early stages of learning should produce movement segments that appear to be almost aimed, with each movement unit appearing to be aimed in a slightly different

\(^4\) That is, the variation operator is intended to produce the smallest possible cost change in the schema. The stochastic search or gradient search will then establish where in this new space the best action lies. This approach keeps the number of actual variation operators needed to a minimum. This simplifies the system considerably. It also minimizes the possible objection that variation operators increase the dimensionality of the task to be learned.

\(^5\) Another variation operator that we are eager to try, but have not as yet, is the crossover operator that genetic algorithms find so useful. This is operator 8 in the figure. To incorporate this variation into our system requires modifying the relevant data structures to support a population of trunk sequences of CSEs. That is, to more closely resemble genetic algorithm we need a schema instance to be not simply the best sequence of CSEs found so far, but instead to be the best \(N\) sequences of CSEs found so far. Such changes are under way.
direction. In time, of course, the boundaries between units should blur, as all the movement units take on a similar direction in space.

The advantage of learning in clusters of CSEs is that (1) minimizing jerk and other cost functionals by search within the unit should move things in the right direction at a dimensionality that (2) makes credit assignment (say by local approximation of the cost surface) a problem of temporarily lowered dimensionality. This approach also makes it possible to find better regions to search by elaborating a movement unit and seeing how much change is created in the search. If only a little change occurs, collapse it back to 1 CSE, and search elsewhere. While mutating clusters may be beneficial for some problems, we found that mutating symmetric pairs (or clusters) of CSEs, i.e. symmetrically in time, to be particularly helpful for learning movements with a symmetric velocity curve.

Finally, each of the variation operators mentioned needs at least one parameter as argument. For example, the split operation needs to know which CSE to split. Similarly, combine needs to know which two adjacent CSEs to combine, and of course the crossover operation needs to know where to cut and splice the crossover. None of these parameters specify the action vector explicitly. That modification is always left to stochastic search. However, the value of the variation parameters can have significant effects on the learning performance at the task. Consequently, the variation sequencing and learning algorithms presented in this chapter always distinguish a (Split-nth \( n_1 \)) from (Split-nth \( n_2 \)) for \( n_1 \neq n_2 \).

4.2 Practise strategies
4.2.1 Informal description of practise strategies

The reason for the two classes of operators introduced in the last section has to do with the way they are used for getting the learning system unstuck when it is unable to make any further progress, and with the way in which the system learns and generalizes methods of resuming progress. Every time a schema instance is retrieved from the database and executed, prior to the execution either a search operator or a variation operator is selected and applied to the schema to make some modification to it. Most often the modifications are subtle changes that do not drastically alter the trajectory of the sensori-motor device. There are occasions where a search operator will make a substantial change in a torque or equilibrium position, such occurrences are infrequent, but important since they provide opportunities for jumping out of a local minimum. By far, the most frequent operations are search operations, which are mostly small changes to the action vectors of a subset of the CSEs in the schema. As we mentioned earlier, search operations are explorations of an action space of a temporarily fixed basis of actions. It is when the search is no longer producing progress, as determined by the cost functionals associated with the task, that a change of the search space may be appropriate. Variation operators are used for this purpose, and so their use is considerably less frequent than search operators. Typically, a variation operation may occur once for every hundred or so search operations. Consequently, storing the sequence of variations tried for a given schema can be an efficient way to capture the history of what the system did to solve the problem. We will return to this point shortly.

The notion of variation operators came into existence during our early attempts at getting the basic adaptive loop to work on the reaching task. The first version of the
system used schemas with several hundred CSEs and TD as the means of doing temporal credit assignment. Only stochastic search and gradient descent were available for search, with no variation operators. The basic adaptive loop did not show any signs of convergence with a few thousand iterations. It was immediately clear that several hundred CSEs, many of which were varying randomly, was not likely to converge without something to focus the search.

The first thing that comes to the mind of any mathematically trained individual in situations like this is to try to determine what simplified circumstances might produce a system that could converge, and to prove the convergence by some kind of inductive approach. It was "immediately obvious" that the system should start with 1 CSE per schema, at least for the arm task, minimize its functionals for this CSE, and then split the CSE into two CSEs, each of half the duration of the first, and proceed to mutate them independently, and continue by induction on the length of the schema. This suggested the following simplified and conditional application of variation operators.

Informal description of a practise strategy for the reaching task

(1) Start with 1 CSE.
(2) Iterate around basic adaptive loop modifying the 1 CSE stochastically, until a sustained local minimum occurs with the cost functionals.
(3) Split the longest (in duration) CSE into two of half duration, but each with the same action vector.
(4) Iterate around the basic adaptive loop modifying each CSE stochastically, either simultaneously or according to some pattern such as round robin, until a sustained local minimum occurs with the cost functionals.
(5) Pick from among the longest (in duration) CSE to split and go to 4.

Informal description of a practise strategy for learning saccades to moving targets

(1) Start with an initial schema of 3 or more CSEs.
(2) Saccade to a moving target using the topographic map for stationary targets, since the moving target map will be empty.6
(3) When progress levels off, try to combine two CSEs into one, or drop a CSE within the sequence. Do not drop off the last one.
(4) Continue improving accuracy with stochastic search.
(5) If progress levels off, try looping back to (3)
(6) Fine adjustment could occur by "discovering" that the linear controller works well if appended to end of schema.

Such ordered, conditional applications of variation operators to get the search process unstuck when it reaches a plateau or local minimum we will refer to as a practise strategy. An important characteristic of our variation operators is that in most, but not all cases, the

6 This presupposes a sensitivity to moving targets, which infants clearly have, though they are not yet able to track moving targets well. Thus, a moving or shimmering object will elicit a saccade, but the saccade will only be on-target if the target hasn't actually moved much.
schema after the variation application computes nearly the same function as the schema before the variation application. This can only be true of variation operators that increase the dimensionality of the space searched. Variation operators that reduce the dimensionality necessarily change the schema more drastically.

The example practise strategies presented above were constructed "by inspection." In chapter 5, we will provide evidence that these practise strategies are effective and produce much more efficient learning than search alone. The remaining sections of the chapter describe a second level of the learning architecture, in addition to the basic adaptive loop of the previous chapter, which is capable of discovering and generalizing practise strategies specific to each task.

4.2.2 Informal discussion of human practise strategies

Motivation for this extra layer of process comes primarily from informal observations about human learning and problem solving.

First, people do use learning strategies, such as study habits, training procedures, etc. It is our hypothesis that sequences of variation operators are the lowest level equivalents to training procedures. The motivating idea is that in the process of trying to solve a problem, when we try something that works, we want to consider trying it again, the next time we are stuck on a similar problem.

Another motivation comes from the possibility that using induction on the length of the schema might be a principled way of improving the search results for many problems.

Still another motivating factor was that we wanted to be able to construct schemas with <sensation-action-outcome> elements from several topographic maps arranged in a sequence (later, perhaps a tree with branch conditions). We needed an operation for performing the juxtaposition, and a criterion for when to consider using another resource, and a principled way of remembering when such sequences work, and when they do not. Each of these are aspects of practise strategy learning.

Finally, it was anticipated that the pattern of search might be useful to capture and reproduce for future movements. Thus, approximation is not the only contribution toward generalizing what is learned when a successful solution to a sensori-motor problem is found.

4.3 The variation learning model and its primary data structures

4.3.1 Introduction

One way of looking at our proposed distinction between search operators and variation operators is that the search operators tend to find the best sequence of actions to take within a space of possible actions. On the other hand, the variation operators tend to help the system find better action spaces to search. We also hinted that saving the sequence of variation operators, and saving information about the conditions at the time of application could be an efficient way of saving the history of the system's pattern of search and practise with each schema instance. We will refer to this history of the pattern of practise as an event history. The event history will be defined in the next section. The notion of event history is not exactly the same notion as that of practise strategy. However, each event history contains information for constructing a practise strategy. A
practise strategy is an interactive program by which the system trains itself. The distinction will be clarified when the notion of practise strategy is treated in detail later in the chapter. By way of introduction, the idea is to capture significant events having to do with the system's progress at the task in an event history for each schema instance. Each event history will contain some information about variation operators that helped the system learn the task more quickly, as well as applications of variation operators that did not help the system make learning progress. Assuming that there is a relation between the structure of the task, and variation applications that improve the task, there will be a way to infer a practise strategy from the collection of event histories. If an effective practise strategy can be inferred and confirmed from a relatively small number of schema instances, then the practise strategy can be applied to new schema instances. Thus, learning a practise strategy while beginning to form a topographic map may greatly enhance learning the remainder of the topographic map. This learning enhancement is in addition to the enhancement that occurs by interpolating new schema instances from the existing schema instances already in the map.

4.3.2 Event histories
Events associated with learning progress

There are two aims in choosing a way of representing the history of progress-related operations and events: (1) to include those variables and operations that are clearly relevant to monitoring learning progress, and (2) to keep the amount of storage space needed for the event history to a minimum. The storage requirement is especially difficult because there is no obvious way to remove the requirement that there be 1 event history per schema instance.

Intuitively, we sought to represent only those events of significant change in some progress-related variable(s). In addition, we are including the variation applications themselves, since it is the relationship between these operations and learning progress that we wish to monitor. Consequently, individual search operations are not considered events of change. Rather, they are considered to be the repetition of an unchanging process. On the other hand, any time a search operation generates a mutation, that improves the performance of the schema instance beyond the best performance achieved so far, it is considered to be an event of significant change. The search operation itself is not recorded as an event, but the change in cost vector is. There is one other category of event that needs to be included. In some instances, such as deleting a CSE from a schema instance, the variation operation can result in an increase in cost of the trunk-sequence by virtue of the change in the vector space of actions that the schema instance is now restricted to. In other words, the best performance achieved by the current schema instance has just gotten worse. The increase in cost is represented as an event that follows such a variation operation, rather than considering the two as one event. These observations lead to the following definition.

Definition 4.3: An event is a tuple of data corresponding to one of two types of occurrences of change in the system. A variable change event corresponds to the change of maximum achieved value of a progress-related variable (or minimum value achieved for a cost). A variation application event corresponds to the application of a variation
operator. In addition to the type field indicating that the event is one of the two
described types, there is a time field indicating when the event occurred in schema-
relative time. Schema time is measured in how many times the schema instance had been
executed when the event occurred. Other fields include fields for the name of the variable
that changed, the change in value, or if the event is the second type, the variation that was
applied, and the credit assigned to this variation. There are bookkeeping fields as well.

As examples of events, the system always monitors changes of cost functionals,
including the weighted average of the all the cost functionals of the task. All variations
can occur as events as well. This includes effective changes in temporal mesh, such as
with split and combine operations, effective changes in spatial mesh, such as when the
upper and lower search limits are altered, and changes in search strategy, such as
switching from stochastic search to gradient descent.

**Event histories organize learning events chronologically**

The information that the example events contain is available after the schema
instance has been executed, and after all the cost functionals have been computed, but just
before the schema is returned to the database (see Figure 3.3). Events do not necessarily
occur with every schema execution. In fact, schemas may be mutated on the order of a
hundred times without either a variable change event or a variation application event
occurring. When an event does occur, however, it is immediately pushed onto the event
history for that schema instance. To return for a moment to the schema instance as a
tuple, the event history is a data structure that is stored as a field of the tuple, as denoted
below.

\[ S_i = \langle \text{CSE-tree}_i, \text{trunk-sequence}_i, \text{search-set}_i, \text{search-parameters}_i, \text{event-history}_i, ... \rangle \]

From the description so far an event history will be a sequence that looks
something like Figure 4.3.

... CE\(_3\), CE\(_7\), VE\(_{22}\), CE\(_{42}\), CE\(_{53}\), VE\(_{65}\), CE\(_{68}\), CE\(_{77}\), CE\(_{83}\), VE\(_{102}\), ...

**Figure 4.3**

In the figure, the subscripts indicate the practise trial of the schema at which the
change occurred. For example, CE\(_3\) indicates a variable change event at the 3rd time the
schema instance was executed. Since CE\(_3\) is an event, it is a tuple of data, including
which variable changed and what the magnitude and direction of the change were.
Similarly, VE\(_{22}\) indicates that a variation application event occurred at the 22nd execution
of the schema. The corresponding event tuple contains information about which operator
produced the variation and what the known effects are on the cost functionals of the task.
According to our definitions, the practise trials not represented in the history correspond
to search operations that did not result in, say, cost improvements in the task beyond the
best cost achieved so far. Thus, trial 6 did not result in improved performance of the
schema instance, nor did the trial limit the best performance achievable. The 6th trial would have had to have been a search operation that produced a mutation whose resulting costs were not as good as the schema instance without the mutation (i.e. the trunk CSE sequence).

**Credit assignment for variations within event histories**

In this section we will describe an algorithm for credit assignment that is applied to variations and events within a single event history. In a later section we will describe a generalization of this approach that will apply among event histories. For the moment, assume that we have some mechanism for determining when to apply a variation operator and which variation operator to apply. As can be observed from Figure 4.3 the variation applications are relatively sparse compared with search applications or variable change events. In addition, the consequences of applying a variation operator, such as splitting 1 CSE into 2, can be quite far reaching. In many cases, the variation operator is changing the dimensionality of the search and minimization task. For these two reasons, any method of temporal credit assignment that we choose should not use discounting of cost attribution over time that is too fine grained. For example, if VE_{22} in Figure 4.3 is a split variation, and if a cost change event, with delta_cost_{42} = -0.1, occurs 20 trials later, at CE_{42}, then if we discount this cost change by G_{trial_count} \times delta_cost_{42}, the credit assigned to the variation is only -0.012, using a discount factor of G = .9. This seems unrealistically low, especially if no additional variation operators had been applied since the split. If, in addition, the split variation had not been applied until the system had been stuck at a plateau for quite a while, then we would expect to credit nearly all of the delta_cost_{42} of -0.1 to the split variation.

Instead, we can view the event history as a sequence of actions, namely the variations, with the cost decreases and increases between variations summed as a single cost change outcome. This gives the variation aspect of the learning problem a simpler structure. Now, the discounted cost change becomes G_{variation_count} \times delta_cost_{42} = G^0 \times -0.1, and the split variation does indeed get credit for the cost change. Note that variation_count is the number of variation applications between the variation getting the credit computation and the specific cost change being credited. Pursuing this example, VE_{22} would also get the credit for CE_{53} at G^0 \times delta_cost_{53}. On the other hand, the credit assigned to VE_{22} for CE_{68} would be G^1 \times delta_cost_{68}. This gives us the following formula for temporal credit assignment to variation operators.\(^7\) Let the practise trials at which variation application events, \{VE_i\}, of event-history\(_h\) occurred be \(i^0, i^1, ..., i^h, ...\).

Then the credit assigned for variable, \(x\), to variation operator VE\(_h^j\) is

\(^7\) This formula for the assigned credit will be amended to handle branching in the event history, which will be introduced in the next section. The formula will undergo a final change to accommodate multiple event histories, when generalized practise strategies are discussed.
\[ \text{Credit}_{\text{VE}_j}(x) = \sum_{(\text{VE}_j \cap \text{CE}_i)} \left[ \sum_{i \in \text{CE}_i} G^{j-1} \times \text{delta}_\text{cost}(\text{CE}_i, x) \right] \]

Equation 4.1

G is the discount factor. 0 < G < 1. Thus, for a variation event, credit for a distant delta_cost is reduced by the factor $G^n$, where n is the number of variations between the variation to be accredited and the cost change in question. This latter discounting has the desired property that it gives credit to an enabling variation for distant outcomes.

The state between variation events

Events are usually thought of as transitions. If this is so, then what are the states between variation events? The state of the learning system between variations can be regarded as that which does not change between variations. A lot of things could be included in this but there are two alternatives that could be appropriate in this context. For the first alternative, recall that variation operators, by changing the action space spanned by the trunk sequence of CSEs, move the system to different search regions. From this point of view, the state between variation transitions could be defined as a region spanned by the action components of the trunk sequence of CSEs, and perhaps also limited by the spatial (i.e. action space) search scale upper and lower limits. In other words, the specific temporal and spatial mesh and bounds between variations could be taken as an important part of the state definition. Also part of the state could be other aspects of search that are held fixed between variations, such as the choice of search algorithm, the search pattern, etc.

An alternative approach to state representation would be to treat each variation sequence as a separate state. This choice might make it very difficult to recognize different sequences as equivalent. Both of these alternatives produce a finite, though changing, state space at any moment the system is running. However, the second alternative is less desirable both from a combinatorial point of view, and from a point of view that seeks the simplest, most general variation sequences by recognizing equivalence classes among the sequences.

To return to the first alternative, the temporal mesh is part of the "state" and is determined by the onset times and durations of the actions of CSEs in the trunk sequence. It should be mentioned that action vector values (and sometimes time and duration values) are not discretized during search, but kept within allowable regions. When variation operators change the spatial scale and mesh, or temporal scale and mesh, these regions are changed by discrete amounts (e.g. a duration is split in half, or doubled). Thus, the possible settable regions of time and other quantities have discrete boundaries. Consequently, there exists a finite collection of variation "states" at any time, even though at the task level search operators move action vectors through a closed and bounded continuum. In addition, these variation level states, though finite in number at any moment, can increase in number without any specific bound. New states can be added, for example, as more CSEs are added to schema, or as CSEs are halved in duration.

To simplify the discussion, we will assume that the temporal scale and mesh determine the current state of the learning system with respect to a specific schema.
instance. More specifically, if a schema instance has \( n \) CSEs, with action onset times of \( t_1, \ldots, t_n \) and durations of \( d_1, \ldots, d_n \), respectively, then the state of the learning system for that schema is just the sequence of time intervals \( t = \{(t_1, t_1+d_1), \ldots, (t_n, t_n+d_n)\} \). If the ending time of each CSE coincides with the starting time of the next, then the state can be regarded as just \( t_1, t_2, \ldots, t_n, t_n+d_n \). In order to use states defined this way for temporal credit assignment or approximation or simply to compare the states of two schemas, we need to define a metric, \( \rho \), on \( X \otimes X \), where the set of possible states is \( X \). This, of course, will make \( (X, \rho) \) a metric space. Note that a simple function from \( X \otimes X \) to \( R^1 \) such as

\[
\rho(t,u) = [(t_1 - u_1)^2 + (t_1+d_1 - (u_1+e_1))^2 + (t_2 - u_2)^2 + (t_2+d_2 - (u_2+e_2))^2 + \ldots + (t_n - u_n)^2 + (t_n+d_n - (u_n+e_n))^2 + (u_{n+1})^2 + \ldots + (u_m)^2]^{1/2},
\]

with \( u = \{(u_1, u_1+e_1), \ldots, (u_m, u_m+e_m)\} \) also in \( X \) and \( m > n \),

**Equation 4.2**

does not satisfy the definition of a metric. Recall that \( \rho \) is a metric if it has the following properties:

i. \( \rho(x,y) \geq 0 \);

ii. \( \rho(x,y) = 0 \) iff \( x = y \);

iii. \( \rho(x,y) = \rho(y,x) \);

iv. \( \rho(x,y) \leq \rho(x,z) + \rho(z,y) \).

Properties i, ii, and iii are easy to verify for equation 4.2, but iv fails. For example, suppose we 3 sequences,

\[
z = \{(1,2), (2,3), \ldots, (n, n+1)\},
x = \{(2,3), \ldots, (n, n+1)\},
y = \{(1.5, 2.5), (2.5, 3.5), \ldots, (n + 0.5, n + 1.5)\}.
\]

From equation 4.2 we can compute values of the metric on pairs of sequences.

\[
\rho(x,y) = (2-1.5)^2 + (3.2.5)^2 + \ldots + (n - (n-.5))^2 + (n+1 - (n+1.5))^2 + (n+.5)^2 + (n+1.5)^2
\]

\[
= 2*(n-1)*(.25) + (n+.5)^2 + (n+1.5)^2
\]

\[
= (n-1)*.5 + (n+.5)^2 + (n+1.5)^2
\]

\[
= 2*n^2 + 4.5*n + 2
\]

\[
\rho(x,z) = 2*(n-1) + n^2 + (n+1)^2 = 2n-2 + n^2 + n^2 + 2n + 2 = 2(n^2+n)
\]

\[
\rho(y,z) = 2*n*(.5)^2 = n*.5
\]
Comparing terms shows that
\[
\rho(x,y) > \rho(x,z) + \rho(z,y).
\]

Thus, property iv fails for the definition of \( \rho \) in equation 4.2.

Intuitively, a metric for our search spaces should be able to give reasonably similar values to sequences of slightly different lengths, but where the alignment of each sequence is taken into consideration in the differences.

A metric that will work is as follows. First assume that \( t_1, \ldots, t_n \) and \( u_1, \ldots, u_m \) are sorted in increasing order. Also assume, for simplicity, that \( t_i + d_i = t_{i+1} \) and similarly for \( u_i \) then define two functions \( T(x) \) and \( U(x) \) based on sequences \( t \) and \( u \) as follows.

\[
T(x) = t_i \text{ when } t_i \leq x < t_{i+1} \text{ for } i < n, \text{ and} \\
T(x) = t_n \text{ when } t_n \leq x < t_n + d_n \\
T(x) = 0 \text{ otherwise.}
\]

Similarly,
\[
U(x) = u_i \text{ when } u_i \leq x < u_{i+1} \text{ for } i < m, \text{ and} \\
U(x) = u_m \text{ when } u_m \leq x < u_m + e_m \\
U(x) = 0 \text{ otherwise.}
\]

then a metric for \( X \) is given by

\[
\rho(t,u) = (T(x) - U(x))^2 \, dx.
\]

**Equation 4.3**

Though it is quite easy to prove that this is a metric on \( X \), we will omit the proof for brevity.

We are still not quite done with the discussion of state. Different ways of getting to the same state will produce very different learning rates, though it is certainly true that the state, as defined, determines the best performance that the schema instance can achieve. However, the schema that produces this performance will not, in general, be unique. The issue is that at the variation level the system is trying to find the shortest sequence of variations that gets the system to a "good" state. However, if the system finds that a 100 CSE sequence is the best state, then why not start with 100 CSEs, rather than starting the schema at 1 CSE, and proceeding inductively? In this example, the actual number of search operations required to reach a specific cost value would be greater when starting at 100 CSEs vs. starting with 1 CSE.

How is it that the system can find the inductive variation sequence when it is not the shortest sequence in numbers of variation operators? One answer is that the system is biased in a number of ways to discover variation sequences that require fewer practise trials. For example, if it should try starting with 100 CSEs in a blank schema instance and
varies all 100 action vectors at once, the likelihood of actually discovering the 100 CSE solution in this manner is extremely low. Thus, it will tend, for reasons to be discussed shortly, to give up on this approach and try another. The system does not stay in a state that does not support learning progress in the task above a certain rate. It is this assumption that the task should be learnable at a certain rate that eliminates the need to search the entire state space exhaustively. The tendency to keep progress advancing at the base level balances out the bias of finding the shortest sequence of variation operators that "solve" the minimization problem to a certain cost level.

The actual choice of variation operators can help in this regard. For example, if the system must always increase or decrease the number of CSEs in a schema by 1 CSE, rather than being allowed to jump by N CSEs, then the system will be forced to explore a wider range of states.  

Thus, while the criterion of achieving a certain level of performance in the shortest time is not reflected in the credit assignment algorithm explicitly, it is reflected in other mechanisms that bias the system toward wider, though not exhaustive, search. It is also important to note that this example, where the number of variation operators is anti-correlated with the number of trials, is more the exception than the rule.

To conclude, this notion of state between variation events is consistent with our choice to sum the cost changes between variation events, treating them as a single value. We will return to this notion later, when we discuss how to extract one generally effective variation sequence from many event histories. Note also that our definition of state does not represent anything in the external environment. Instead, it represents part of the system's internal state as it attempts to learn the task. This is in keeping with our orientation toward understanding how an interactive organism is structured internally, as opposed to modelling the environment.

The temporal credit assignment algorithm (discussed later in the chapter) will use this simple definition of state, and maintain its own state-transition information in the database. Thus, an event history represents a path through state space for which the assignment of credit for cost reduction is known.

Branched organization of events in event histories

Events have two organizations in the event history. First, there is the chronological order of events in the event history, where the time of occurrence of an event corresponds to the number of times the schema instance has been executed when the event occurred. With the second type of organization, events in an event history have a tree structure. In this tree structure alternative branches from a node correspond to returns to the same point in the learning process, but trying a different sub-sequence of variation operators instead.

---

8 The system is not restricted in such an extreme way. It can change the number of CSE's by different amounts, usually between about 1 and 10.
Figure 4.4: The diagram contains one event history at seven stages of growth, labelled 1 through 7. 3’ and 7’ are identical to 3 and 7, respectively, except that they are drawn in a way that makes the branch structure apparent. At stage 1, a variation operator, V1, has just been applied, but has not produced any cost gains. At stage 2, the lack of progress results in the events following V1 (inclusive) being terminated and marked as an ineffective attempt. Ineffective attempts become branches in the data structure with a NODE inserted at a knot before the branch. At stage 3, the search is continued with variation, V2, and its aftermath. At stage 4, the V2 sequence is terminated as ineffective, and becomes another branch of the search process. Stage 5 is another attempt with variation V3. Stage 6 shows that V3 was ineffective. However this time the frustration affect is much higher. Consequently, the NODE for the new branch is pushed farther back, absorbing variation V0. The search is resumed with variation V4.
For example, if a variation application event occurs, and if progress is slowed or halted for a considerable period of time after this event, then a branch is formed. Figure 4.4 shows the steps of the branching process. The branch extends from the variation application event to the last event in the history. A node event is inserted just prior to the variation application event. The last event is labelled as a terminal or leaf of the tree. The state of the history at the variation level and the state of search at the base level are replaced with the state of the learning system just prior to the ineffective variation application. A new branch is then begun by connecting the node to the empty space at the end of the event history where the next new event will go.

For this algorithm to work, the credit assignment algorithm and several predicates must be chosen carefully. There is a predicate for deciding when to terminate the current event sequence, and a predicate for determining how far back up the tree to resume exploration. Ideally, the trunk of the tree corresponds to a better variation sequence than would have occurred by letting the event history continue unbranched. We will discuss these predicates in the later sections on goal and effect processes of the second level system. For now it is sufficient to point out that the predicate for deciding when to terminate a branch is based on the rate of cost functional reduction per practise attempt. This rate is a measure of how fast (or slowly) the system is learning. When it is below a threshold value then the most recent stretch of the event sequence is terminated to form a branch. The critical question is how far back in the event history the branch should go.

To determine how far back the branch should go (i.e. where the node should be made), there are two considerations. The first is how far back in event history time this period of slow learning can be traced. The second consideration is how much branching and backing up the system has already done. When learning progress has been slow for a very long time, then a quantitative measure corresponding to "frustration" at the learning progress will reach a high level. When the frustration is at a low level, then the system will look backward in time as long as the learning rate is slow. As soon as it finds a period of time where the learning rate is higher, it will stop and put the node on the boundary between the period of faster learning and the period of slower learning. This, of course, requires another threshold having to do with what rate of learning is considered "good enough to keep" at the current frustration level. However, as the frustration level increases, and more variations have been tried and withdrawn into branches, then threshold of "good enough" progress is raised. As a result, protracted periods of slow learning push back even farther in time, eventually removing very early variations that may have produced some short term learning progress, but should now be attributed to the long term lack of progress.

This kind of selection from the event history of a more effective subsequence could have been accomplished by selecting and piecing together "good" subsequences from several unbranched event histories, instead. However, in the course of implementation we found that putting some ability to back up and try again into a single event history ultimately simplified algorithms for inferring a practise strategy from what are ultimately better quality event history trunk sequences.
Definition 4.4 An event history is an ordered tree of events that is associated with a schema instance, where there is only one unterminated branch, the trunk of the tree, where the events in each branch are arranged chronologically, and where each terminated branch consists of a sequence of events, a search, that was not able to sustain a threshold level of learning progress in the task. Each terminated branch represents a pattern of search which was aborted, and after which the state of the system was returned to the state of the node before the branch began. Consequently, the lowest cost path through the event history tree is the trunk.

4.3.3 Section Summary

In this section we have introduced the notion of event history as a chronology of events that record the learning progress of the base level system. We have also introduced some very simple mechanisms for assigning credit for fast progress or slow progress to variations in the event history. Implicit in the discussion has also been a search process that works with the credit assignment mechanism to find variation sequences that sustain learning progress.

The event histories together with the rudimentary credit assignment and search processes are sufficient (demonstrated in simulation) for the system to "learn", for example, to construct a sequence of split variations in the arm movement task. The pattern of splitting will also tend to split CSE's evenly (i.e. split the longest ones first). However, we have only outlined enough mechanism so that a practise strategy can be learned for a particular schema instance. This practise strategy is embedded in the event history as those variations that have not been withdrawn into branches (i.e. remain in the trunk). However, we have not presented any mechanisms for using such a practise strategy to guide further learning. Nor have we presented any mechanisms for determining whether such a practise strategy is applicable across the entire topographic map (at the base level), or whether it is applicable across some subregion. In other words, we have not yet presented a means of generalizing from individual event histories, those common subsequences that are sufficient to sustain learning and applicable across regions of the topographic map. Such mechanisms are the topic of the next section.

4.4 Variation adaptive loop and its relation to the basic adaptive loop

4.4.1 Overview of the variation level

Recall that in the basic adaptive loop (Figure 3.3) there is a box labelled "Variation Operators". The variation level of the architecture supplies the operators for the actions that this box applies to the schema instance. As discussed earlier, these may be either search operations or variation operations. There are two different ways in which the variation executive (the interpreter at the variation level) can perform this selection. One is from a predefined, script-like practise strategy, which is essentially a recipe or program of search operations and variation operations in a fixed sequence. The other way in which the variation executive can supply operators of change is to learn a practise strategy for the task, while the base level is learning the task. This latter method is possible because the first few steps of the practise strategy can be learned as it is learning the first few schema instances. These steps can then be applied to begin learning more schema instances, during which time the system is also learning more of the practise strategy.
Thus, the process of learning a practise strategy exploits the horizontal decalage\(^9\) inherent in the trial-and-error learning process for the entire topographic map. In this way, the system depends on being able to learn different schema instances at different times, or at least a few ahead of the rest.

**4.4.2 Variation Adaptive Loop**

**Introduction**

Two observations can help toward understanding the reasons for the design and organization of the variation adaptive loop. The first observation is that the variation level works in relation to the same cost functionals as the base level. There are, however, certain biases in the system toward finding faster ways of minimizing these cost functionals. The second observation is that the event history, one of which is associated with each schema instance, captures everything significant that happened to the schema instance in the course of trying to learn the base level task (e.g. the moving target task, or the reaching task). The goal for the variation level of the system is to promote progress in learning the base level task, and during this process to learn a practise strategy for the base level task.

As a learning system itself, the variation adaptive loop has the same structure as the basic adaptive loop. A key difference, however, is that the variation adaptive loop does not interact with the external environment. Instead, it interacts with an internal environment, the schema instances and affects of the base level.

**Brief overview of the variation adaptive loop**

Consider Figure 4.5. As with the basic adaptive loop (Figure 3.3) there is a goal process, a database containing the event histories (a kind of variation schema), a new level and type of variation operators called *meta variation operators*, an *internal* analog of senses and motors, which are the base level schema instances and cost functionals, and an affect process specifically associated with the goal for this level.

As with the basic adaptive loop, we will describe the process of executing one iteration around the variation adaptive loop. To begin, when some large number of trials have gone by without any further minimization of the cost functionals, the goal process of the variation level kicks in and tries to take some action that will keep the learning process of the base level moving. This learning process is currently running on behalf of a particular schema instance.

---

\(^9\) This was Piaget's term for different levels of skill, that are displayed in the child's performance, for tasks that are the same or similar in the mental operations they require.
Figure 4.5: As with the basic adaptive loop, the goal and affect processes define the task and supply criteria for success and improvement at the task. The topographic map, or local database, contains the event histories and generalized variation schemas associated with the goal. The variation schema is both a program for performing interactions as initiated and maintained by the goal process, and the repository of all experiences at attempting the 2nd level task. In this case, the variation schema is operating on 1st level schemas, with the outcome of the interactions determined by how the changed 1st level schemas learn the basic task. The meta variation operators change variation schemas toward improvement and greater generality.
For the variation level, the sensory input (for a database query) is the schema instance and associated measures of progress in the base level task, which are by our definition the affect measures of the base level. What the database holds are the event histories, which are essentially templates consisting of trees of \(<\text{affect-context}, \text{2nd-level-action}, \text{outcome}, \text{2nd-level-affect}>\) tuples.\(^{10}\) Each event history contains this information as a subsequence of variation operators. The primary database index for the event history is the schema instance.

Once the appropriate event history is retrieved from the database, the system needs to determine what action should be applied to the schema instance.

Among the actions the variation level can take is to apply a variation operator to the schema instance, or to undo a branch of the event history. Undoing a branch of the event history also changes the schema instance. It returns the schema instance to an earlier state, just before the ineffective variation was applied.\(^ {11}\) For example, if a CSE had been removed by the variation operator, now known to be ineffective, then undoing the variation entails putting back into the schema instance the CSE that had been removed. After re-establishing the state of the schema instance and event history, the system needs to find another variation to replace the ineffective one.\(^ {12}\) We have already discussed the process of undoing a branch of the event history, so we need not repeat it here. The process of finding a variation to apply to the schema instance involves one of two choices. The first choice is to use stochastic search in variation space. The second choice entails retrieving nearby event histories, and using the information in the event histories to suggest a next variation to try. Extracting a variation from these nearby event histories is a process that combines or modifies them, in such a way that a longer event history is produced, which can supply the needed variation operator to modify the schema instance and resume learning progress.

To clarify the second choice, the schema instance of the base level that is responsible for the activation of the variation level constitutes part of the "sensory input" to the variation level. The event history corresponding to this schema instance has been retrieved from the database. However, the process of finding a variation operator to apply requires information not in the nearest retrieved event history (the one that is indexed directly by the schema instance), though possibly in nearby event histories.

If there are very few other event histories in the database, then the next variation would most likely be selected by stochastic search from among the full set of variation operators. Otherwise, a meta variation operation can be tried. The retrieved sequence of variations in the event history can undergo one of a set of genetic-like operations. These genetic-like operations can either transform the original event history to have a longer

\(^{10}\) These are variation-CSEs. They are analogous to CSEs at the base level in that they are the smallest unit of interaction at the variation level.

\(^ {11}\) Recall that undoing a branch of the event history means determining that an earlier variation in the history was ineffective and backing up the state of the learning system at both levels to the moment just before this variation was applied.

\(^ {12}\) This is assuming that the conditions indicate not to continue with search operations on the base level schema instance.
(1) **Extend-from-Matching-Portion-to-End (VS, n)**

\[ VS = \text{XYZZXY}, \text{ a sequence of variation operators} \]
\[ n = \text{length-of-match, } n \in \{1,2,3\ldots\} \]

Looks for a match of the last \( n \) elements of the sequence to some earlier portion of the sequence. When it finds a match, it takes the next element after the match and copies it to the end of the sequence.

For example, given a sequence is XYZZXY, a next element, is desired. The element found is in boldface font.

for \( n = 0 \), XYZZXY extends to XYZZXYY
for \( n = 1 \), XYZZXY matches with XYZZXY and extends to XYZZXYZ
for \( n = 2 \), XYZZXY matches with XYZZXY and extends to XYZZXYZ
for \( n = 3 \), XYZZXY has no match and returns nil

(2) **Crossover-at-nth (VS\(_1\), VS\(_2\), n)**

Exchanges the subsequence of variations at the \( n \)th position between two variation sequences.

For example, a crossover at \( n = 5 \) produces
\[ VS_1 = \text{XYZZXY} \rightarrow \text{XYZXXEF} \]
\[ VS_2 = \text{ACBADEF} \rightarrow \text{ACBADY} \]

As another example, a crossover at \( n = 6 \) produces
\[ VS_1 = \text{XYZZXY} \rightarrow \text{XYZZXYF} \]
\[ VS_2 = \text{ACBADEF} \rightarrow \text{ACBADE} \]

(3) **Extend-from-Matching-Portion (to-VS\(_1\), from-VS\(_2\), n)**

\[ \text{to-VS}_1 = \text{the sequence to be extended by 1 element} \]
\[ \text{from-VS}_2 = \text{the sequence to be searched for a match} \]
\[ n = \text{the length of the search pattern} \]

Searches the sequence from-VS\(_2\) for a match to the last \( n \) elements of to-VS\(_1\). When it finds a match within from-VS\(_2\), it takes the next element after the match and copies it to the end of to-VS\(_1\). For example,

\[ n = 2 \]
\[ \text{to-VS}_1 = \text{XYZZXY provides the pattern XY} \]
\[ \text{from-VS}_2 = \text{XRABC has a match to the pattern XY, thus} \]
\[ \text{to-VS}_1 = \text{is extended to XYZZXYR} \]

**Figure 4.6:** A set of meta variation operators that is designed to find ways to extend a sequence of variation operators by one more variation. In each case, this is accomplished by using the last \( n \) elements of the sequence, that requires the single operator extension, as the search pattern. Either the sequence itself or a second sequence is searched for a match to the pattern. If a match is found, the next element after the match in the searched sequence becomes the desired extension.
sequence of variations, or transform the original event history in combination with nearby event histories to generate a longer sequence of variations that borrows more from the nearby event histories. Observe that this assumes that the set of genetic-like operations can operate on sequences of unequal length. See Figure 4.6 for a small set of operators that accomplish what we need.\(^{13}\) These operators resemble the crossover operator used in genetic algorithms.

Applying one of these meta variation operators will produce a variation that can be applied by the "action system" of the second level. The variation so produced will then be executed with the stuck schema instance as its argument. The result of the execution will be a modified schema instance, a modified event history, possibly altered cost functionals, and some saved system state.\(^{14}\)

The affect process of the second level monitors the outcome of the variation application. This outcome is not available until the modified first level schema instance has itself been executed, and the outcome of the base level task is available. The second level affect process thereby interprets the outcome of the first level process. In fact, the second level affect process may require many executions of the schema instance before it has a clear verdict on the new variation application. The scale of time of activity in the second level is on the order of 10's or 100's of iterations of schema instances. However, the affect process will keep track of learning progress, storing its assessments in the event history between variation applications.

In the next sections, we will look at each of the components of the variation adaptive loop in more detail.

**Goal process of the variation adaptive loop**

The goal process of the variation adaptive loop is activated only by progress-related variables (affects). The possibility of the goal becoming active occurs at most once per movement attempt (i.e. once per base level schema execution), though the actual frequency of goal firing is much less than this. The rate of goal activation at this level depends on how certain predicates and thresholds are set. For settings we typically use, the goal fires roughly once every 50 to 100 movement attempts.

An explicit termination predicate for the 2nd level goal process is not necessary. Once the goal is activated an action or bundle of actions is taken, which will modify 2nd and 1st level data structures. If the goal process is successful, then changes in values of progress-related variables will reflect this and the goal will not be re-activated on the next cycle.

\(^{13}\) N.B. each of these genetic-like variation operators can be synthesized from the four primitives: SEPARATE, RECOMBINE, SHIFT and ALIGN. Note also that keeping pointers to the source sequences of successful crossovers around for a little while can significantly improve the likelihood that effective sequence extensions will be retried before other variation operations or search operations.

\(^{14}\) Some state must be saved in case the learning experience with this schema instance is discontinued after a while.
<table>
<thead>
<tr>
<th>Affect threshold (level 2)</th>
<th>Interpretation (level 1)</th>
<th>Actions (level 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>very positive affect</td>
<td>(learning task ++)</td>
<td>• put Δcost into event history</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• update credit assignments</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• mark high confidence variations</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• update generalized variations</td>
</tr>
<tr>
<td>moderately positive affect</td>
<td>(learning task +)</td>
<td>• put Δcost into event history</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• update credit assignments</td>
</tr>
<tr>
<td>neutral affect</td>
<td>(learning task 0)</td>
<td>• goal remains inactive</td>
</tr>
<tr>
<td>moderately negative affect</td>
<td>(learning task -)</td>
<td>• update credit assignments</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• search for variation to apply</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• put variation into event history</td>
</tr>
<tr>
<td>very negative affect</td>
<td>(learning task --)</td>
<td>• update credit assignments</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• terminate unsuccessful branch in the event history</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• update generalized variations</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• search for variation to apply</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• put variation into event history</td>
</tr>
</tbody>
</table>

**Figure 4.7:** This is the variation (2nd) level goal process. It is triggered by any one of the non-neutral affect thresholds. The affect at this level is a real-valued function of the level 1 cost functionals and time. This one affect number can be interpreted as a success/frustration value (positive/negative value) which interprets the learning progress of the base level task. Any monotonic function of costs and time will do that is increasingly positive when Δcost/Δtime is increasingly negative, and negative when Δcost/Δtime ≥ 0. The labels very positive, moderately positive, moderately negative and very negative correspond to thresholds of the monotonic affect function as the value of the function moves away from zero. Changing these thresholds in relation to the function, obviously alters the rate at which the second level system attempts to correct the learning progress of the first level system.
The following events activate the 2nd level goal process. Figure 4.7 contains a diagram of the triggers and consequent processes. First are cost change events. These are events that decrease the cost functionals below the best value so far. A less common form of cost change event occurs when the costs increase in such a way that they cannot be lowered further, that is, without undoing the variation that pushed them up. When cost change events occur, the goal process reacts by asserting a cost change "event" into the event history, and by performing all credit assignments within the event history that result from this change.

Second are events that credit a variation with sufficient cost reduction to exceed a high confidence threshold. The variation is labelled with a high confidence mark, which makes the variation part of a stable subsequence of variations in its event history. An additional consequence is that the variation can become part of a generalized variation schema as well.

Third are occasions when the 2nd level frustration affect reaches a moderate threshold suggesting that a variation should be found to apply at the 1st level. Such occasions trigger numerous second level processes to find a suitable variation to modify the stuck base level schema instance. The details of these processes are discussed in the remaining sections of this chapter. See also Figure 4.8.

The fourth and final type of event corresponds to a higher threshold for the 2nd level frustration affect, which suggests that some of the recently applied variations have not been effective. This case initiates processes that seek to attribute the high level of frustration to likely previous variations and to withdraw these variations into branches within the event history.

With respect to the fourth case, which is triggered when the 2nd level frustration affect reaches a very high threshold, a variation is withdrawn into a branch because the credit assignment process has determined that the variation was ineffective.

Affect process of the variation adaptive loop

There is one progress-related variable which is used for two affect poles (i.e. positive and negative) by the variation level. The two affect poles correspond to success at the base level learning task, and to frustration with progress at the base level learning task. Increasingly positive values of this one variable correspond to success, while negative values correspond to frustration. Figure 4.7 shows how the second level goal process uses the progress-related variable to guide the learning process.

The progress variable at this level is a real-valued function of the level 1 cost functionals and time. Any monotonic function of costs and time will do, if it is increasingly positive when Δcost/Δtime is increasingly negative, and negative when Δcost/Δtime ≥ 0. The affect levels in Figure 4.7 are very positive, moderately positive, moderately negative and very negative. These levels correspond to thresholds of the monotonic affect function as the value of the function moves away from zero. Changing these thresholds in relation to the function obviously alters the rate at which the second level system attempts to correct the learning progress of the first level system.
Search progress is waning so that a variation is needed to alter the schema instance.

Figure 4.12 contains an exhaustive list of comparisons between possible variation sequences in an event history and corresponding variation sequences in the associated gvschema.

Cases A and B are clearly cases where the event history is shorter than the gvschema, and matches it well. Consequently, these are cases where the variation should be selected by following the gvschema. While Cases A and B represent only 25% of the cases in the figure, they tend to be the most frequently encountered cases in practise.

Cases C, D, and E are marginal cases, for which the current gvschema could be used, although there may be a better match in the database. If there is no such match, then defaulting to the current gvschema is the first variation to try.

Cases F, G, H indicate that the event history has followed all the suggestions of the gvschema, and they have worked, but the gvschema is too short to suggest the next variation to try. In these cases it is probably best to search the database for another gvschema, though it is not likely that one will be found because of our restriction that gvschemas can't be proper subsequences of other gvschemas (at least not with the same initial element). Consequently, genetic-like search and stochastic search will most likely be beneficial for these cases.

Cases C, D, and E match their assigned gvschema marginally, but could conceivably match another gvschema better. If such is the case, then use the variation from the better matching gvschema.

Cases F, G, and H are all cases where the currently associated (by G) gvschema is not long enough to suggest a good match (unless a periodic pattern from earlier in the sequence is used). Since it is unlikely that another gvschema could match better, genetic-like search should be considered here.

Here is the search option list. They are ordered by priority. That is, when a variation (that hasn't been tried before) is needed, try (a), then (b), then (c) ... until one is found.

(a) Search first in the associated gvschema.
(b) Else search for a direct match in other gvschemas (reassign if necessary)
(c) Else use genetic-like search on gvschemas (as discussed earlier),
    sort returns by cost at requested state and pick best.
(d) Else use genetic-like search on event histories,
    sort returns by cost at requested state and pick best.
(e) Else use a mutation (stochastic search).

Figure 4.8: This is the search algorithm that corresponds to the cases enumerated in Figure 4:12. The search algorithm uses the relation, G, between event histories and meta variation schemas that is maintained by the algorithm in Figure 4.11. The search is initiated by the goal process when a variation is needed.
Variation Level Database: Event Histories and Generalized Variation Schemas

In an earlier section we defined a data structure that we called an event history. Event histories record special progress-related events in the learning experience of schema instances. Event histories can be regarded as schemas themselves, where the variation operators in the sequence specify a kind of action, and the progress-related variable information is a kind of context for the subsequent action. Since event histories record actions that have already happened, it may not be obvious how an event history can be a pattern for future action. In this section, we will explore the question of how effective patterns of variation operations in event histories can be determined and re-used both to guide the learning of future, developing schema instances, and sometimes to guide the later learning of the schema instance that gave rise to the event history.

Consider, once again, the minimal set of operators in Figure 4.6. It is not hard to see how such operators can be used. For example, Extend-from-Matching-Portion-to-End (VS, 0) just copies the last variation of the sequence, VS, onto the end of VS. As a variation sequence, this amounts to repeating the thing you did last. If you have a sequence VS₂ that is longer than VS₁, you can make a sequence that takes the extra elements of VS₂ and appends them to VS₁ by using Crossover-at-nth (VS₁, VS₂, length-of (VS₁)). Finally, Extend-from-Matching-Portion-to-End, except that it searches for a match in a different sequence than the one containing the source pattern.

The following algorithm is sufficient for good portions of variation sequences to migrate to nearby event histories. Figure 4.10 contains a concise statement of the algorithm. The idea is relatively simple. Assume S is a schema instance for which the learning process is stuck, and assume that EH₁ is the event history associated with S. Let V₁, ..., Vₙ be the n variation operator subsequence of EH₁. Then we would like the algorithm to generate possibilities for the variation, Vₙ₊₁. These candidates for Vₙ₊₁ can be tried during the course of many practise trials of the schema instance S. The genetic-like operations summarized in Figure 4.6 look for candidates for Vₙ₊₁ by considering the K last variations of EH₁, Vₙ₋ₖ, ..., Vₙ, and searching nearby event histories to EH₁ for subsequences that match Vₙ₋ₖ, ..., Vₙ. If there are no matches, K is reduced by 1 and the search is tried again until a match is found. The very worst case would result in Vₙ₊₁ = Vₙ.

There are several obvious limitations of this algorithm. One limitation is that it does not explicitly use the search state induced by the variations. As a result, it will tend to do a lot more search than it needs to. A related limitation is that it has no way of recognizing that minor differences in variation sequences can be ignored. Thus, instead of finding a small number of variation sequences for training a topographic map, it will find many sequences and miss general patterns among them. Though the first limitation could be fixed, the algorithm would still not have any mechanism that pushes toward finding relatively few, generalized variation sequences. We pursued a supplementary approach, which will be discussed shortly.
### Base Level

<table>
<thead>
<tr>
<th><strong>Goal Process:</strong></th>
<th><strong>Affect Measures:</strong></th>
<th><strong>Senses:</strong></th>
<th><strong>Motors:</strong></th>
<th><strong>Variation source:</strong></th>
<th><strong>Generalization:</strong></th>
<th><strong>Schema structures:</strong></th>
<th><strong>Credit assignment:</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Onset at target appearance.</td>
<td>Task cost functionals.</td>
<td>Visual target tracking</td>
<td>Torques to actuators</td>
<td>Variation Operators</td>
<td>GRBF approximation centered</td>
<td>Trees of CSEs (schema instances)</td>
<td>By surprize affect</td>
</tr>
<tr>
<td>Offset at grasp or timeout.</td>
<td>Short term frustration.</td>
<td>Tactile sense</td>
<td>Equilibrium position control of actuators</td>
<td>Search Operators</td>
<td>at CSEs as data points</td>
<td>Trunk of tree is best interactive path</td>
<td>By approximation of cost surface</td>
</tr>
</tbody>
</table>

### Variation Level

<table>
<thead>
<tr>
<th><strong>Goal Process:</strong></th>
<th><strong>Affect Measures:</strong></th>
<th><strong>Senses:</strong></th>
<th><strong>Motors:</strong></th>
<th><strong>Variation source:</strong></th>
<th><strong>Generalization:</strong></th>
<th><strong>Schema structures:</strong></th>
<th><strong>Credit assignment:</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Onset at progress plateau.</td>
<td>Combined task cost functional.</td>
<td>Event history.</td>
<td>Variation Operators</td>
<td>Meta Variation Operators</td>
<td>Generalized variation schema relation</td>
<td>Trees of variation-CSEs (event histories)</td>
<td>Modified temporal difference</td>
</tr>
<tr>
<td>Offset at progress resumption.</td>
<td>Long term progress-related frustration.</td>
<td>Progress-related variables.</td>
<td>Search Operators</td>
<td>Genetic-like and stochastic search</td>
<td></td>
<td>Trunk of tree is best interactive path</td>
<td>Presumed minimum rate of progress</td>
</tr>
</tbody>
</table>

**Figure 4.9:** The table shows the structural similarities between the base and variation levels. The structural similarity is only an isomorphism in the weak sense of having similar components and functions (homomorphic doesn't quite describe it either).
 Needless to say, there are many details that we have omitted. As we have just described the variation adaptive loop, it works very much like the basic adaptive loop with a change of senses and motors. In fact, the database at the variation level has a similar tree structures as the schema database of the base level. Thus, we are well on our way to constructing a level whose structure is similar to that of the basic adaptive loop.

Figure 4.9 shows the isomorphism. In this approach, the event histories, which are copied into the database as variation schemas, correspond to the schema instances of the base level. In fact, there is a 1-1 and onto correspondence between schema instances and event histories. While this approach (two level system with the simple algorithm) does work, the only mechanism that could push the system in the direction of reducing the number of distinct variation sequences for the entire topographic map would be some kind of approximation (based on a metric devised for the state space underlying the variations) with the variation-CSEs as data. Initially, we tried such a 2-level system, which did learn successive refinements of the temporal mesh for the arm task.

We decided to look for an algorithm that does a bit more work toward finding a more general kind of practise strategy (general for the topographic map to which it is applied). The improved algorithm tries to construct generalized variation sequences using the event histories as data. It is not fully a 3rd level of the system. Rather, it is analogous to the role of GRBF approximation at the first level, which makes a kind of generalization by interpolation and limited extrapolation possible. However, to perform this generalization on event histories, additional data structures are needed. Among the added data structures are what could be called generalized variation schemas. Generalized variation schemas have a similar structure to event histories, though their structure is more compact, and they are less numerous. In a sense, they add a thin layer of data to the topographic map. The description of these architectural extensions and the extended algorithms begins with the next section. However, before we can discuss the algorithms of search and generalization for these additions to the variation level, we need to give a brief description of the database for this level.

**Generalized variation schema data structures**

The event histories do not comprise the entire database at the 2nd, variation level. As we have already mentioned, there are data and index tables constituting an implementation of generalized variation schemas, which we will refer to as *gvschemas* for short. Generalized variation schemas are like event histories, except that much of the variable change information and some of the branching information is removed. However, the CSEs composing the generalized variation schemas are the same variation-CSEs as in the event histories, though with increasingly generalized contexts. Event histories retain the memory of what was tried and what the outcome was (i.e. the search tree for the particular schema instance). Gvschemas retain only the successful, high confidence subsequence of variations from event histories. In addition, each gvschema contains information about the event histories that support it, as well as context information for each variation operator, which is a generalization of the context of corresponding

---

15 Strictly speaking, the base and variation levels are not isomorphic, nor are they homomorphic in any set theoretic or group theoretic sense. We are using the term loosely to denote that the two levels are structurally similar in components, functions and data structures.
variations in the supporting event histories. By support, we mean those event histories with a subsequence of effective variations that match the sequence of variations of the gvschema. Thus, gvschemas contain information about their own generality and applicability for the task, as well as how confident or reliable they are.

While gvschemas are constructed from portions of the earliest successful event histories, their primary purpose is to provide a general template or guide for variation sequencing as new event histories are grown. It is not assumed, however, that there will be one such template for an entire topographic map. It may be that as new regions of schema space are explored, the earliest gvschemas will not apply or at least will need modification. The mechanisms outlined below will handle such divergence as well.

Event histories capture the information associated with searches through variation space, and the tree structure within the event history represents both successful and unsuccessful attempts in their branches, with a growing trunk constituting a relatively successful, or high confidence path through the tree. For now, assume that we have a predicate that will allow us to mark such a path from the root of the tree along a sequence of successful variations. While it is possible for no such path to exist, it should be rare that this happens.

A generalized variation schema is a sequence of variation-CSE (variation operators with some of the search context data). This is also analogous to the relationship between a schema and a CSE. Since the purpose of keeping generalized variation schema data structures is to find the most effective, and most general variation sequences, we will impose some constraints on relations between event histories and gvschemas. Figure 4.10 shows a picture of a hypothetical instance of the relation, which may clarify this discussion of constraints.

The first constraint is that generalized variation schemas must be unique up to sequence of operators. Event histories, on the other hand, are not necessarily unique with respect to the sequence of variation operations. This implies a many-to-one relationship between event histories and gvschemas.

The second constraint is that each event-history points to at most one corresponding generalized variation schema. This would be the gvschema whose sequence of variation operators contains the high confidence subsequence of variations in the event history.

The third constraint is that there exists an index table such that each gvschema points to all event-histories that point to it as their containing variation sequence.

The fourth constraint is that an event history's high confidence subsequence cannot be longer than the gvschema it points to. However, it is nevertheless allowed for event histories to be otherwise longer than gvschemas. Event histories can have recent variations whose credit assignment is, as yet, unclear. Thus the recent variations are not marked as high confidence. Such event histories can remain assigned to shorter gvschemas, until the recent variations get marked as high confidence. When this happens, the corresponding gvschema must be extended to contain the new variation(s).
Event Histories

Event-History-1 ...-VE1==VE2== ...-VE1==VE2==VE5== GVSchema1
Event-History-2 ...-VE1==VE2==
Event-History-3 ...-VE1==VE2==VE5==
Event-History-4 ...-VE1==VE2==VE5==VE7----
Event-History-5 ...-VE9==VE10--VE13-- ...-VE9== GVSchema2
Event-History-6 ...-VE9==VE10--
Event-History-7 ...-VE9==VE10-----
Event-History-8 ...-VE9==VE10----VE19--
Event-History-9 ...-VE99----VE90----

Generalized Variation Schemas

**Figure 4.10:** The diagram above shows an example of the relation, G, between event histories and gvschemas that satisfies the constraints listed in the text. These constraints produce a relation where each event history is associated with at most 1 gvschema. Note that in this example Event-History-4 is likely to have been the first event history to have developed. It would have given rise to GVSchema-1. Event histories 1 - 3 would have grown using GVSchema-1 as a template.
For the fifth and last constraint, suppose gvschema$_1$ has $n$ variations in its sequence and gvschema$_2$ has $k < n$ variations in its sequence. Then gvschema$_2$ must not be the first $k$ variations of gvschema$_1$. However, gvschema$_2$ could still be a subsequence of gvschema$_1$, it just cannot be isomorphic to the first $k$ elements of gvschema$_1$. We could make this constraint stronger, but it will do as it is for now. This will at least guarantee that all event histories that start out similarly get mapped to the same gvschema.

These constraints are exhibited by the example in Figure 4.10, showing the expected relationship between gvschemas and event histories. This relationship has been chosen so that the system can explore a range of different variation sequences (as practise strategies), but spot good subsequences relatively easily. How this is done will be discussed shortly. It should be clear from this description that the worst case, from the point of view of finding a practise strategy is a 1-1, onto map of event histories to gvschemas. In which case, every schema instance would have a unique gvsquence associated with it. The best possible case, on the other hand, would be if all the event histories were mapped to one gvschema. That gvschema would be one general practise strategy for the entire topographic map.

Maintaining a relation, G, between gvschemas and event histories

For clarity, we will begin with an example of how generalized variations can grow out of event histories while abiding by the constraints listed in the previous section, and capturing a degree of generality in the variation sequence. The algorithm corresponding to the example is presented in Figure 4.11. There are two parts to the use of generalized variation schemas, the first part is concerned with the construction and indexing of the generalized variation schema data structures, which includes maintaining the relation between event histories and gvschemas. The second part to the use of gvschemas involves using them to suggest variation operations when a schema instance is not making progress learning its task.

Suppose we have several event histories, represented schematically as follows. In this representation the single thickness line segments (usually hyphens) indicate cost change events that are relatively small, double lines (made with equal signs) indicate very good learning progress, and that the preceding variation is credited with at least part of this progress. A plus sign in the path indicates a branch, though we will not be concerned with branches in this discussion. Variation operators are represented with the subscripted letter V.

EventHistory$_1$ \ldots-V$_1$---
EventHistory$_2$ \ldots-V$_2$==
EventHistory$_3$ \ldots-V$_2$---V$_3$==V$_4$------
(1) **Initializations.**

Initially, there are no event histories. Event histories are created with the first task learning events.

Initially, variations are marked with 0 confidence when pushed onto an event history.

Initially there are no meta variation schemas in database. MvSchemas are created as variations in an event history are marked as high confidence due to the temporal credit assignment algorithm.

Initially, \( G(EH_m, MVS_n) = \emptyset \). The many-to-one relation, \( G(EH_m, MVS_n) \), between event histories and mvSchemas is maintained by this algorithm, where for each \( m \), \( EH_m \) appears at most once in \( G \).

A 1-1 correspondence between schema instances and event histories is maintained by level 2.

(2) **Search progress is good & decreases costs so that a credited variation becomes high confidence.**

This change in the variation within an event history drives a change in its associated mvSchemas as follows. Let this variation and event history be \( V_j = EH_m(j) \).

(a) **If** there is no mvSchemas associated with the event history in \( G \),

i.e. \( \neg \exists MVS_n \ni G(EH_m, MVS_n) \),

then search the DB for an mvSchemas that matches the sequence with the new variation,

i.e. search for \( MVS_n \ni \) for \( k = 1, \ldots, h \),

where \( \{i_1, \ldots, j = i_h\} \) are the indicies of high confidence variations up to \( V_j \) in \( EH_m \).

\( EH_m(i_k) = MVS_n(k) \),

and if such an \( MVS_n \) is found, then assert \( G(EH_m, MVS_n) \).

and if no such \( MVS_n \) is found, then create a new mvSchemas with the new variation sequence,

i.e. construct \( MVS_0 \) such that \( EH_m(i_k) = MVS_0(k) \) and assert \( G(EH_m, MVS_0) \).

(b) **If** the associated mvSchemas still matches the high confidence variation sequence of the event history in question,

i.e. if \( G(EH_m, MVS_n) \) and if \( EH_m(i_k) = MVS_n(k) \), for \( k = 1, \ldots, h \), where \( \{i_1, \ldots, j = i_h\} \) are h.c. indicies up to \( V_j \) in \( EH_m \),

then do nothing, except to update the context of this variation in the mvSchemas (see section ???).

**Otherwise** if the event history is consistent with the mvSchemas as far as the mvSchemas goes,

i.e. if \( EH_m(i_k) = MVS_n(k) \), for \( k = 1, \ldots, g < h \), and \( V_g \) is the last variation in \( MVS_n \),

then extend the mvSchemas's variation sequence with the credited variation,

i.e. for \( k = g+1, \ldots, h \) set \( MVS_n(k) = V_i_k = EH_m(i_k) \).

**Otherwise** if the event history has deviated from the mvSchemas,

i.e. if \( \exists g \ni g \leq h \) and \( MVS_n(g) \neq EH_m(i_g) \),

then search the DB for an mvSchemas that matches the new sequence (as in 1.a),

**Otherwise** if search finds no match, then make a new mvSchemas,

i.e. construct \( MVS_0 \) such that \( EH_m(i_k) = MVS_0(k) \) for all relevant \( k \),

and remove \( (EH_m, MVS_n) \) from \( G \), and assert \( G(EH_m, MVS_0) \).

**Figure 4.11:** The pseudocode above is the first part of an algorithm for maintaining the relation, \( G(EH_m, MVS_n) \), between event histories and meta variation schemas. See next page for rest of figure.
(3) Search progress is poor enough that a credited variation is pushed into a branch (3 cases).

(a) If the pushed variation (credited with 0 cost reduction, or worse) was also part of the mvschema, i.e. if $V_j = EH_m(j)$ is a variation that is marked as low confidence and if $EH_m(i_k) = MVS_n(k)$, for $k = 1,\ldots,h-1$, where $\{i_1,\ldots,i_{h-1}\}$ are all h.c. indicies up to, but not including $V_j$ in $EH_m$, and if $V_j = MVS_n(h)$, then the mvschema's variation sequence can't any longer match the event history's sequence, since $V_j$ would need to be high confidence for this, so dissociate the event history from the mvschema, i.e. delete the tuple $(EH_i, MVS_j)$ from $G$, and search for another mvschema that matches the altered event history's sequence, i.e. use search as in (1.a) above, followed by construction if needed, otherwise if the search returns no mvschema, then make a new mvschema.

(b) If the pushed variation is not part of the mvschema (so far), so that there's no conflict, i.e. $V_j \not\in MVS_n(k)$ for $k = 1,\ldots,h$, (it could have been generated before the mvschema grew into present variations) then let the mvschema suggest the next variation to apply, i.e. try $MVS_n(h)$.

(c) If the pushed variation leaves the event history with no high confidence subsequence, then dissociate the event history from the mvschema. I.e. delete the tuple $(EH_i, MVS_j)$ from $G$, (N.B. the event history still has record of what didn't work.) When another variation is needed, try to match the event history with another mvschema that doesn't conflict with the faulty variation. This is following the search definition of (1.a).

<table>
<thead>
<tr>
<th>Notation:</th>
<th>$EH_m(j) = V_j = \text{the } j\text{th Variation of } EH_m$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$MVS_n(j) = V_j = \text{the } j\text{th Variation of } MVS_n$</td>
</tr>
</tbody>
</table>

Figure 4.11 (continued): The pseudocode above is an algorithm for maintaining the relation, $G(EH_m, MVS_n)$, between event histories and meta variation schemas. The algorithm is driven by two categories of events: (1) when a variation in an event history becomes marked as high confidence, and (2) when a new variation is needed for a stuck schema instance. The event history and mvschema tables are indexed by discrimination trees (Charniak and McDermott, 1984) for fast searches.
Now suppose that in EventHistory₁, there is significant progress after V₁, which is attributed to V₁, and that the aftermath can be marked as high confidence. The diagram for the event history then becomes,

EventHistory₁ \[\rightarrow V₁ \rightarrow\]

Suppose that there are no gvschemas in the database that contain the variation V₁. Since we now have a variation that works with some confidence in the context of EventHistory₁, we can create GVSchema₁, which contains V₁, and some context information, C₁. Thus,

GVSchema₁ = \{<C₁, V₁>\}

and the tuple (EventHistory₁, GVSchema₁) is entered into a many-to-one relation, G, that captures the generalization relationship between event histories and gvschemas. If there had been gvschemas in the database, we would have looked for one whose first variation was V₁, and associated EventHistory₁ with it.

Suppose that the relation, G, now contains the following elements.

\{(EventHistory₁, GVSchema₁)
(EventHistory₂, GVSchema₂)
(EventHistory₃, GVSchema₂)\}

where GVSchema₂ = \{<C₂, V₂>, <C₃, V₃>, <C₄, V₄>\}. Suppose also that EventHistory₂ needs a next variation. Since it is associated with GVSchema₂ in G, the variation V₃ can be used (ignoring context for now). So we have,

EventHistory₂ \[\rightarrow V₂ \rightarrow V₃ \rightarrow\]

Now, if V₃ works nothing special happens. However, if V₃ does not work well and learning stops, as would only be known over a large number of trials. Assuming that we know now, EventHistory₂ would become branched and other variations tried. This could be denoted,

V₃ \[\rightarrow\]
\[\rightarrow V₂ \rightarrow + \rightarrow V₅ \rightarrow\]

One question, of course, is where V₅ came from. If there were a richer assortment of gvschemas, then some genetic-like combination might produce a candidate variation.
However, in this case, V5 would most likely be generated stochastically from the collection of all variations. Once it is determined that V5 is effective for EventHistory2, then the tuple containing EventHistory2 in G, (EventHistory2, GVSchema2), is removed from G. The reason for this is that GVSchema2 cannot any longer be considered a generalization of EventHistory2. Furthermore, if there are no gvschemas in the database other than GVSchema1 and GVSchema2, then EventHistory2 would remain unassociated with another GVSchema until V5, or some other variation, causes learning progress to resume. At such a time, if there are still no matching GVSchemas, one will be created. Call it GVSchema3. Thus, G would be,

\[
\{(\text{EventHistory}_1, \text{GVSchema}_1) \\
(\text{EventHistory}_2, \text{GVSchema}_3) \\
(\text{EventHistory}_3, \text{GVSchema}_2)\}
\]

Finally, suppose a schema instance is relatively unpractised and requires its first variation. Its event history, say EventHistory4, will not be associated with a gvschema in G. One way to choose what to try would be to consider each GVSchema, and to choose not only on the basis of the minimum cost achieved for so many trials, but also to consider the number of relation tuples it is in, and to consider how well the context matches the generalized context. As an implementation of these considerations, we sort the GVSchemas by a simple weighted sum of these three quantities, and use the GVSchema with the largest weighted sum. We call this weighted sum the gvschema’s reliability with respect to the ith variation in the gvschema.

\[
\text{reliability}(\text{GVSchema}_j, i) = \\
w_1 \times \text{cost\_reduction\_credited}(V_j) + \\
w_2 \times \text{cardinality\_of}(\{(\text{EventHistory}_j, \text{GVSchema}_j)\}) + \\
w_3 \times |\text{context\_of}(\text{EventHistory}_k, i) - \text{context\_of}(\text{GVSchema}_j, i)|
\]

The algorithm corresponding to the discussion of the last few paragraphs is given in Figure 4.11. The algorithm assumes the existence of two predicates on event histories or progress-related affects. The predicate that corresponds to the condition of step (2) of the algorithm is the "very positive affect" predicate referred to in Figure 4.7. The predicate is applied both to the progress-related affect and to a specific variation within an event history to determine whether the variation should be marked as one of high confidence. High confidence indicates that there is evidence from patterns in the values of the cost functionals over time that everything in the trunk of the event history up to the specified variation constitutes a variation sequence that will be kept as the trunk.
Figure 4.12: A, B get next variation from mvschema, H contributes a variation to the mvschema, and C, D, E, F, G, H require search beyond the mvschema.
<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| (0)  | $S$ = schema instance that is stuck and requires a variation.  
|      | $k_{\text{min}}$ = smallest pattern length to be used.  
|      | $k_{\text{max}}$ = largest pattern length to be used.  
|      | $N$ = maximum number of event histories to retrieve. |
| (1)  | Retrieve the $N$ nearest event history neighbors to $S$, call them $EH_1, \ldots, EH_N$. |
| (2)  | Sort $EH_1, \ldots, EH_N$ by distance from $S$ (any euclidean metric on DB keys will do). Assume that the list is relabelled so that $EH_1$ is the closest event history to $S$. |
| (3)  | Generate new extended sequences of $EH_1$, by applying the following meta variation operators:  
|      | (a) collect 
|      | for $1 < j \leq N$ and for $k_{\text{min}} < k \leq k_{\text{max}}$  
|      | $\text{Extend-from-Matching-Portion} \ (EH_1, EH_j, k)$ |
|      | (b) collect when not nil 
|      | for $1 < j \leq N$  
|      | $\text{Crossover-at-nth position} \ (EH_1, EH_j)$ |
|      | (c) collect when not nil 
|      | for $i = N$ down-to 0  
|      | $\text{Extend-from-Matching-Portion-to-End} \ (EH_1, i)$ |
| (4)  | Consider only those products of crossover operations that contain the variation subsequence of $EH_1$, and sort them by likely outcome, which is approximated by the outcome from their source event histories. Push these variations onto the search queue $S$. |
| (5)  | If a queue of successful crossover-like operations, patterns, and their source event histories is kept for likely reuse, then the system can learn from its positive crossover experiences. |

**Figure 4.13:** This algorithm constitutes a genetic-like search for a single variation operator to extend the sequence of variation operators in an event history. The event history, $EH_1$, corresponds to the learning history of schema instance, $S$. This search is activated when learning has stopped progressing for schema $S$. 
The second predicate corresponds to the condition of step (3) of the algorithm. This predicate tests for the opposite condition of the first predicate. That is, it checks to see whether there is very negative affect associated with an event history. If so, the current trunk variation sequence of the event history should be terminated and turned into an unsuccessful branch.

Searching the database and making use of generalization

The second part of our discussion of gvschemas involves how to use gvschemas to suggest variation operations when a schema instance is not making progress learning its task. There are two possibilities overall. First, if the gvschema associated with the current event history represents a longer sequence of variations than the n variations of the event history, but contains this n variation subsequence of the event history, then a simple match and alignment of the sequences will yield the n+1st variation to try. Second, when the event history no longer matches the gvschema it is associated with, then some search is necessary, either to find a better matching gvschema, or if none is available, performing a genetic-like search to construct a better template to follow. Figure 4.12 and Figure 4.8 contain a sketch of the algorithm for this process. Figure 4.12 contains an exhaustive set of ways in which an event history can match, or fail to match its associated gvschema. Figure 4.8 contains the algorithm that the system uses to search for a next variation to try.

As can be seen from the figures, the cases where search is required are precisely those cases where the event history has grown beyond or away from the gvschema. We have used 2 methods for searching for new sequences or subsequences: a genetic-like search (as discussed earlier in this section, and in Figure 4.13), and stochastic search. We will also try temporal difference methods at some time in the near future. The first 2 methods are applicable to many situations, but are especially well suited to extending sequences beyond the longest sequence found so far. Temporal difference methods are more suited to finding more efficient sequences from among those that have been explored, by eliminating potentially redundant operations. Thus, they minimize the number of trials required to attain a certain level of cost performance.

Sorting gvschemas for best one to grow an event history along

Many of the steps of the search algorithm in Figure 4.8 return a list of matches, each of which may have a credited cost change associated with it. Thus, there may be the task of sorting the candidate gvschemas to determine the best one (or several) to try next. The credited cost change alone is not obviously a good measure for sorting since it needs to be normalized for how many trials (how much time) contributed to the cost change, and it needs to be normalized for the number of event histories that support the gvschema. By support, we mean how many event histories are related to the gvschema in G. Performing both of these normalizations provides a good measure for sorting the returns of the search algorithm. Thus, a gvschema with many event histories in its support will tend to be followed by new event histories as they develop, since it will always sort to the top of the list. Some randomness can be added to this process so that there's exploration beyond one gvschema, even if it works extremely well.
Using temporal difference methods for hypothesizing better gvschemas

The methods discussed so far will tend to discover workable variation sequences, though there are still going to be different sequences that are nearly equivalent in their effect. The question naturally arises whether there is a way to search for the simplest, most efficient sequences. While genetic operations will tend to find these in time, there is a more direct approach, namely temporal difference methods (TD). Since TD techniques have become quite popular, we have described a simplified use of them in chapter 6. We will sketch a use of these techniques at the variation level here.

When there are lots of event histories in the variation level database, and there is considerable difference among them, then there is a great deal of state/variation/cost-change data that could be used to find shorter gvschemas via a temporal difference approach. What must be done, however, is to make the assumption that there is an optimal practise strategy, which of course may not be true for a given problem. If this assumption is made, then applying TD methods is straightforward. TD is applied to the state/variation/cost-change information, which produces an estimated valuation assigned to each state. To convert this information into a sequence of variations, one must guess or compute a sequence of variations that will take the system from its usual starting state, to the best state known from the data so far, and which passes through the high valuation states. This new variation sequence can then be put into the database and given a high priority so that it will be tried by new event histories.

Assuming some familiarity with TD techniques, the problems that proponents of this approach tend to study have a finite state representation, where all states are visited in the learning process, and solutions to the problem are minimum time solutions. This means that a cost functional that conflicts with a minimum time functional is not guaranteed to converge with TD. Furthermore, it must be possible to assign values to specific states. This assumes that being in a state has a certain value independent of the path to that state. From a history of state-action-state-payoff information, a policy is approximated which is a map from states to actions. This map is much easier to construct if the state machine of the problem is known so that if an action, \( a_t \), is taken from state, \( s_t \), then the subsequent state, \( s_{t+1} \), is known without observation. As a last constraint, any TD problem, though it is an infinite horizon problem, must reach some designated "goal" state or target state which sometimes is the only state of the problem with a positive payoff.

Our base level task does not satisfy any of these constraints. The state machine is not finite, and it is not known in advance of the problem. Furthermore, the minimum jerk constraint (or any minimum energy constraint) conflicts directly with the minimum time nature of the TD approach. There is no \textit{a priori} value that can be assigned to a state independent of what happens before or after the state in the minimum jerk task. Thus, TD is not at all appropriate for our base level task.

Our variation level task, however, satisfies many of the TD requirements. Our simplified notion of variation state, that we presented at the beginning of this chapter, is finite at any moment in the event history. Although, it can exceed any bound placed on the number of states that is set before the learning algorithm is run. Furthermore, variations as actions take the system to predictable next states. In addition, the minimum time nature of the TD method is consistent with the variation level goal. On the other
hand, the variation level learning algorithm cannot visit all states of the implicit variation state machine, nor is there any end state of maximum payoff.

The latter 2 observations are not a problem for our event histories and associated mechanism. It is assumed that a "good" state will be one in which a certain amount of cost-reduction-per-unit-time occurs. Thus, there is no need to exhaustively search the state space. A path through state space is abandoned if it does not show any cost reduction for a sufficiently long period of time. This also obviates the need for any "final state" or end state with a special reward. Moving the costs ever lower is the only thing the variation level seeks.

To summarize, there are several advantages to using TD at the variation level rather than at the base level. These advantages include:

1. the use of multiple cost functionals, some of which would otherwise conflict with TD;
2. the approach does not require an exhaustive representation of the state space;
3. nor does it require exhaustive search of the state space;
4. our approach can rely on approximation to fill in the gaps in the state space in regions of interest;
5. the overall approach (at both levels) focuses search within promising regions of the state space.

What TD adds to the genetic approach is a means by which the system can assign credit for what got it unstuck, which may not otherwise be easy, because the effects of a variation can be delayed in time. In addition, TD may have some tendency to remove redundancy from a sequence of variations that may, for example, loop back to a previous state unnecessarily.

4.4.3 Section summary

To summarize, we began with a notion of event history that allowed us to save important information regarding the system's progress in learning a task. Since search is performed at the event history level, the system tends to gather many different variation sequences that could be used as practise strategies. It may be that the event histories all fall into a smaller number of equivalence classes. The gvschema level provides some inertia in the search model of the system so that it will find fewer different sequences. However, there still needs to be more of a mechanism to eliminate redundant subsequences within these equivalence classes, which are very likely to arise. TD promises to be effective for this last task, though we have not tried it at the variation level.

Taking a broader view, what this system does is to use a combination of stochastic search, and genetic-like search (and TD in the future) to solve a minimization problem that has numerous local minima, not to mention discontinuities. Gradient descent alone would not be effective. However, our application of genetic search techniques is novel in that the system tracks the effects of non-mutation operations, and attempts to retain sequences that work so that such sequences can be re-used on new parts of the task space. This reuse of what we call a practise strategy supplements the use of interpolation and very limited extrapolation to capture generalities discovered during the learning process.
4.5 Summary, observations, and hypotheses
4.5.1 Summary and parallels between the levels

In this chapter we have extended the base level architecture of the previous chapter to another level, the variation level. The variation level provides the means by which alterations are made to schemas of the first level. These alterations are a form of search that ultimately lead to more successful schemas. This search, we hypothesize, is a possible mechanism for what Piaget termed accommodation.

The base level of the architecture, what we called the basic adaptive loop, and the variation level, presented in this chapter, have many parallel structures and are two applications of adaptive loops. In the case of the base level, the senses and motors are the boundary between the system and the outside world, the site of the interaction. The variation level interacts with the base level of the system. Its analog of senses are the cost values, or any progress related variables and affects. Its analog of motor actions are the variation operators. Thus, the site of the interaction is really with the schema instance and variables associated with search and progress regarding the schema instance. There is a higher level analog to the base level frustration effect. At the higher level too, the system times out if it does not make sufficient progress, and tries another approach. Changing the approach involves backing up prior the offending variation and resuming the search process from the restored state. Thus, the variation level interacts with an "inner world", but is otherwise analogous to the base level.

4.5.2 Why variations do not make the "curse of dimensionality" worse

One possible objection to the variation approach we have sketched out in this chapter is that it increases the dimensionality of the search space with respect to the base level task. It is, of course, true that the total dimensionality of the search space is increased. However, we will list some arguments that suggest that this increase in dimensionality is "forgivable" because (1) it is possible that the dimensionality of the search problem at any given moment may be smaller than the dimensionality of the search problem without variations, (2) a great deal more experience for learning is available to the variation level than the amount of experience available to a single schema instance, and (3) the cardinality of the increased search space can be small, compared with the cardinality of $R^1$, for carefully chosen variation sets. Here are some specific observations and arguments in these directions.

(1) The search through variation space occurs for all trajectories, so there are lots of opportunities to test particular variation operator sequences.
(2) Critical variation operators need only fire when progress has levelled off. Thus, the added dimensionality of exploration only occurs when and if it is needed. That is, the times when the added dimensionality is searched are sparse compared with search of the usual task dimensions.
(3) Once an effective practice strategy has been learned, then learning the remainder of the trajectories for the topographic map can be quite rapid. There is a time after which variation space is no longer searched and the resulting strategy is applied "for free."
(4) Variations tend to both restrict and focus the search, so that the dimensionality of the search at any moment is smaller than the full dimensionality of the task.
(5) Variations restrict the search in order to determine whether there exists a path through variation space that will train the sensorimotor system faster than unrestricted, unconstrained search.

(6) Conceivably, variations can exploit symmetries in the task.

(7) Genetic-like variations are not easily trapped by local minima. For some tasks, this is an advantageous type of search, whether or not variations are learned as a sequence.
Chapter 5 Results of the simulation experiments

5.0 Introduction

This chapter contains descriptions and results of simulation experiments that were proposed in chapter 2. The context of developmental research and motivation for most of these experiments was also presented in chapter 2. In addition, several experiments were performed simply to demonstrate that the learning system, as described in chapters 3 and 4, actually works.¹ It should be mentioned that our primary emphasis in this chapter is on developmental processes in infancy and how simulations can explore models of these processes. There will be little or no emphasis on engineering or performance issues unless they bear some relation to developmental issues.

By way of introduction, we have mapped the relationships between the experimental hypotheses presented in chapter 2 and the corresponding simulation experiments performed in this chapter. These relationships are contained in the following chart. You may wish to refer back to this chart while reading about the simulation results.

Chapter 2

Hypothesis Set 1
(1) Reaching develops out of reflexes.

(2) Reaching develops during growth.

Hypothesis Set 2
(1) Reaching with a minimum-time constraint shows few movement units.

(2) Reaching with a minimum-jerk constraint shows more movement units, which reduce in number during development.

Hypothesis Set 3
Reaching with a minimum-time constraint shows greater linearity than reaching with a minimum-jerk constraint.

Chapter 5

Experiment 1: A 1 equilibrium point reach is essentially a reflex swipe. Most of our simulations start this way.

Experiment 2: Shows that with 4-8 movement units, learning can continue even with rapid growth of the limbs.

Experiment 3: The minimum-time constraint shows 1 movement unit in our experiments. We did not use the minimum-time constraint with high stiffness or growth. When we try such an experiment we may find that it resembles the developmental data.

Experiment 2: During fast growth and high joint stiffness, inflections appear in reach trajectories that have the appearance of movement units.

Experiment 3: This is the main result of the experiment.

¹ To some extent the experiments in sections 5.1.1, 5.5.1 and 5.6 are of this nature, though they are motivated by issues in development as well.
Chapter 2

Hypothesis Set 4
A speed-curvature relationship exists in infant reaching movements.

Hypothesis Set 5
As stated in (Clifton, et al, 1991), infants do not have to see their own hands, either to learn reaching or to accomplish reaching after it is learned.

Hypothesis Set 6
The sequences of low magnitude saccades that infants make have an explanation in the developmental processes that guide the learning of saccades.

Hypothesis Set 7
Smooth pursuit eye movements can develop from saccadic tracking movements with a few additional developmental mechanisms.

Chapter 5

Experiment 4: During fast limb growth, high joint stiffness, equilibrium point control and with the minimum-jerk constraint, inflections spontaneously occur in reaching trajectories. These inflections show a speed minimum exactly at the point of greatest curvature. If there is a relatively small number of movement units, further practise removes the inflections.

Experiment 1: Demonstrates that visual siting, proprioceptive guidance and tactile termination are sufficient to learn the dynamic control of reaching movements. These movements show adult quality (low jerk and ultimately high linearity).

Experiment 5: While we have not explored all the hypotheses fully, we have demonstrated that a learning strategy that starts with a moderate number of saccades per grasp, and develops a strategy that reduces the number over time is efficient for both stationary and moving targets.

Experiment 5: A mechanism is demonstrated by which a sequence of tracking saccades to a moving target can be learned. The additional steps by which smooth pursuit may develop out of the saccade sequence remain to be hypothesized.

5.1 Experiment 1: The basic reaching task.

5.1.1 Basic results
Figure 5.1 shows the basic reaching task where a simulated visual system provides the location of a target object in visual system coordinates. In this case, the coordinates are of the form (x_{retina},y_{retina}), which denote the projected position of the object onto a simulated retina. When a target is available, the system will reach toward it. If the target remains available, the system will keep practising as long as it is able to improve its performance. For purposes of comparison, after each reach the arm was placed back in the same initial position. It is not necessary, of course, to use the same initial position, but we were trying to get a sense of how much practise would be necessary to produce reasonably effective reaches, in the absence of nearby schemas available for interpolation.
Figure 5.1: An assortment of movement trajectories. The upper left trajectory corresponds to a randomly placed equilibrium point before the target had been hit for the first time. Notice that the arm bounces off the upper arm’s counterclockwise limit of rotation. This produced a movement of very high jerk. The upper right trajectory depicts the first time the randomly swung arm hits the target. Notice that the point of contact with the target corresponds roughly to a collision with the forearm. The lower left trajectory shows the arm slowing down and contacting the target both more accurately and with less impact. The lower right trajectory contains an example that has continued to reduce total movement jerk.
Figure 5.2: A reaching trajectory with 4 equilibrium points, displayed as circles, and projected into visual space. The right hand is shown in its resting position after the reaching movement. The dots indicate successive positions of the hand for equal units of time. Each equilibrium point corresponds to the location toward which the hand will go when the controller drives the arm toward a given shoulder angle and elbow angle. The time line above the trajectory trail shows how long the controller is driving the arm toward each equilibrium point. Notice that in most cases, the arm does not pass through the equilibrium point during the time the controller is driving the arm toward it.

Figure 5.3: A reaching trajectory and its corresponding 8 equilibrium points. As in the example above, the equilibrium points are numbered in the order in which they would be activated to generate the trajectory. Notice that the equilibrium points are clustered into pairs. This is the result of an application of the split variation (see chapter 4) to each of a set of 4 equilibrium points (actually 1, 3, 5, and 7) as in the example of figure 5.2 above. Initially, equilibrium points 1 and 2 coincided, as well as equilibrium points 3 and 4, etc. In time, the clustered pairs move apart as the system seeks to minimize jerk.
Success in the reaching task is determined by the tactile location and contact pressure when the arm collides with the target. Any contact with the object terminates the reaching movement. A portion of the arm near the tip is designated the hand, and this is the reference point for the final position error cost functional that was discussed in chapter 3. In other words, when the hand or arm collides with the target, contact locations closer to the center of the hand are considered to have lower final position error than contact locations on the arm, for example. Thus, the accuracy of the reach attempt is determined by simulated tactile senses, not by visual apprehension of the location of the hand as it moves (see also chapter 2 for the developmental studies that motivate this choice). Similarly, the pressure at contact is used as an indirect indication of final velocity. Lower values of the velocity of the arm at contact are considered to have a lower error ($v=0$ is the goal).

Returning to figure 5.1, the top left graphic shows the trajectory of the 2-link, 2-d simulated robot arm corresponding to one randomly situated equilibrium point. The trajectory of the tip of the arm shows a sharp bend where the upper arm bumped into the 90 degree counterclockwise travel limit of the shoulder joint. The top right graphic shows the trajectory when the arm first hit the target. With one equilibrium point, a trajectory can be found that can minimize the final position error to zero. However, final velocity error cannot easily be reduced to zero, and the total accumulated jerk cannot be reduced very much. With two equilibrium points, the final position error can be zeroed and the final contact velocity (or contact pressure) can also be zeroed, but the jerk cost functional (see chapter 3) cannot be substantially minimized. With more equilibrium points, the system can seek minima that most often zero the final position and final velocity cost functionals, but improve the jerk cost only so far, depending on the number of equilibrium points. The bottom left graphic shows such a trajectory guided by 4 equilibrium points. Since most of the jerk is along the axis of movement, this larger component of jerk must be reduced considerably before the transverse component of jerk, which contributes to tip movement (non)linearity, can be reduced significantly. The bottom right trajectory corresponds to such a case where the system used about a dozen equilibrium points to continue to reduce the movement accumulated jerk.

Figure 5.1 shows that when the system can repeatedly practise reaching toward a stationary target, it can begin with one equilibrium point, practise a bit and then try 2 equilibrium points, etc. Thus, it will proceed inductively, increasing the number of equilibrium points in the control sequence as long as it can continue to make better movements. Furthermore, the only visual information needed for the task is the visual location of the target. The learning system did not use visual sighting of the hand for the task. It only needed proprioceptive and tactile sensory information pertaining to the hand. This result resolves hypothesis 5 in chapter 2.

---

2 Recall that jerk is the derivative of acceleration. The total accumulated jerk is a measure of how much the arm changes acceleration during the movement. Minimizing the total accumulated jerk produces a movement that is as smooth as possible.

3 The visual location of the target is not necessarily stationary. The system can perfectly well learn to reach toward a moving target, though we haven't provided any moving target examples for the arm task. We do use moving targets with the head-eye movement tasks later in the chapter.
Figures 5.2 and 5.3 show where the learning system places equilibrium points to generate a movement. For example, figure 5.2 shows the locations of 4 equilibrium points and the trajectory they generate when activated in succession. Each equilibrium point is known to the system as a pair, (shoulder_angle, elbow_angle), rather than as a location in visual space. The equilibrium points are shown as the position of the tip of the hand in visual space for clarity.

Notice that equilibrium point 1 is located just beyond the resting position of the hand, and that it is active for the first 25% of the movement. This means that the equilibrium controller approximates a spring with the equilibrium point as the resting position of the spring. However, the controller switches to equilibrium point 2 well before the arm reaches equilibrium point 1. In fact, the arm never actually gets to equilibrium point 1. The trajectory is thus produced by activating equilibrium points 1, 2, 3 and 4 in sequence with equilibrium points 1 and 2 accelerating the arm toward the target, and equilibrium points 3 and 4 braking the arm so that it comes to rest at the designated target.

Figure 5.3 shows a trajectory and the 8 equilibrium points that generate it. In this case, the sequence of 8 equilibrium points evolved out of a sequence of 4 equilibrium points. Earlier in the learning process, one of the 4 equilibrium points split into two successive equilibrium points corresponding to the same control position, but for successive time intervals of half the length of the initial time interval. This splitting is due to an application of a split variation (see chapter 4 and figure 4.2). Subsequently, another equilibrium point split into 2, until all four were split. After each split, the pair of equilibrium points tend to migrate away from each other as the learning system seeks to minimize jerk. This progressive migration can be seen if the 4 clusters are compared. The clustered appearance is lost with more practise.
5.1.2 Comparison of equilibrium position control vs. direct torque control

As mentioned in chapter 3, we experimented with two methods of controlling actuators with the learning system. With one method the system would learn the function, \texttt{torque(articulator-state, time)}), and with the second method the system would learn the function, \texttt{equilibrium-position(articulator-state, time)}. In the course of running simulations, we noticed that cases of the second method seemed to learn more quickly than cases of the first. To determine whether this observation was true, we ran 10 data sets of learning the \texttt{torque} function, and 10 data sets of learning the \texttt{equilibrium-position} function. Each data set consisted of a practise sequence of 2000 consecutive exploratory movements with the same initial and final position (i.e. the same trajectory was to be learned), but progressing from 1 control structure element (CSE) per movement to between 8 and 12 CSE's by the 2000th movement. In other words, one data set corresponded to 2000 practise trials for one schema instance. The schema instance would start out with a control function consisting of one control structure element (CSE), and through practise the control function would be refined to contain 8-12 CSE's.\textsuperscript{4} Results were assessed by comparing the amount of reduction in accumulated jerk during each movement set (see chapter 3 for a definition of schema, CSE, and a description of the cost functionals).

\textsuperscript{4} The action field of a CSE would be either a torque vector or an equilibrium position vector, depending on whether the torque function or the equilibrium function was to be learned.
Figure 5.4: A comparison of learning performance between movement representations using equilibrium positions (top plot) and movement representations using torque step functions (bottom plot). The equilibrium position controlled movements tend to minimize jerk about an order of magnitude lower than open loop torque functions in the same time period. Data for each plot was from 10 runs, with 2000 movements per run.
Figure 5.4a: Three movements from the equilibrium position control data of figure 5.4. The upper trajectory is a movement whose total accumulated jerk (in relative units, which are the same for all the figures) is 3.04. The middle trajectory has a total accumulated jerk of 1.08, and the bottom trajectory has an accumulated jerk of 0.234. The schemas for these movements contained between 8 and 10 equilibrium points. These movements were learned in 2000 trials each. Notice that the lowest jerk movement is not the most linear. The reason is that the jerk can be lowered orders of magnitude further than any of these movements, and that the largest component of jerk is axial w.r.t. the movement. The transverse jerk, which is correlated with the linearity of the movement, contributes only slightly to the total jerk.
Figure 5.4 shows the results. For each movement set, the lowest jerk that was achieved by the 100th movement, the 200th movement, ..., and the 2000th movement was extracted from the raw data. This data has been plotted as the minimum jerk found so far, vs. n movements tried so far. The pooled 10 equilibrium function data sets appear in the upper plot, and the pooled 10 torque function data sets appear in the lower plot. In the pooled plots the upper "error bar" denotes the minimum jerk so far of the worst data set. The lower "error bar" denotes the minimum jerk so far of the best data set. The trace in between corresponds to the median value of minimum jerk for the pooled data. Both the equilibrium data and the torque data were generated for the comparison and were not selected from a larger data set. Thus, the data in the figure constitute a random (though small) sample.  

From these data it would appear that equilibrium point control has a developmental advantage over an "open loop" control function. That is, it takes fewer practise trials to achieve a "smooth" movement when the control function uses a sequence of equilibrium points.

Figure 5.4a shows some of the trajectories from this experiment. These trajectories were produced at the end of the 2000 practise trials for each of three runs. The schemas producing the movements consisted of 8 to 10 CSE's. In this case the action taken by the CSE is to command an equilibrium position to the controller of the simulated articulator.

What may be surprising about this figure is that the movement with the lowest accumulated jerk is not the most linear. The reason for this apparent anomaly is that it is possible to lower the accumulated jerk by several more orders of magnitude below the best movement in the figure. What we have seen in our data is that the component of jerk along the axis of the movement is quite large compared with the transverse component. Consequently, it is possible for a movement to be very low in accumulated jerk, but still have a noticeable transverse swing. Figure 5.5a shows a more typical case and gives a better feel for how different values of relative jerk correspond to trajectory shapes.

While the trajectories of figure 5.4a are not as linear or smooth as adult 2-d arm movements, they reach their final position with very high accuracy (better than 1%), and their final velocity is quite near zero (to better than 0.1 percent). Furthermore, 2000 practise trials is not a large number for a stochastic algorithm, and probably fewer practise trials than an infant would make in several months of reaching toward one spot in her/his workspace. Thus, the system is capable of producing "useful" reaching movements with a comparatively small amount of practise. This observation may also be true of infant reaching movements. They may not be very linear, but after 5 months, or so, they are effective in getting to the target, and remain so as they improve in linearity.

---

5 Since the search process is stochastic, and the data samples are small, the "learning curves" are far from smooth. The sudden drop of the error bar in the lower curve is due to one data set where the accumulated jerk wasn't improving at all until the stochastic search process found a "good direction" for further minimization. This would engage gradient descent until a minimum is found along the good direction vector.
5.1.3 Section summary

In this section we showed that the learning system discussed in the previous two chapters can learn to make accurate reaching movements with a dynamic 2-d simulated robot arm. Furthermore, even with a relatively short practise period, it can make movements that would be adequate for the task of reaching. The system usually learns to make reaches that have an accurate final position and have a final hand velocity very near zero, in several dozen practise attempts.

We also showed that the learning system is capable of lowering movement jerk significantly, even in 2000 practise trials. Our data indicate that a reduction of total accumulated jerk by 3 or 4 orders of magnitude is typical for this number of practise trials.

Finally, some evidence was presented that indicated a difference in learning performance between the cases where the system is learning a sequence of equilibrium positions for the articulator, vs. cases where it is learning a sequence of torques. Further evidence for this finding will be presented in the next section, where the same comparison will be made when the learning must occur as the arm is growing.

5.2 Experiment 2: Learning to reach while the arm is growing

5.2.1 Discretization of the growth process

We designed the data structures, that we have referred to as schemas, to support the learning of sequential tasks under changing conditions. The most notable changing condition during infancy is growth. In this section we will present several experiments that were performed to see whether our learning system can continue to make effective and improved reaching movements while the arm is growing.

First, we discretized the growth process in the following way. The length of each link and the radius of each link of the arm are independent parameters and can each be increased, in a growth step, by some percentage of their current value. For the next set of experiments, we set the size of the growth step for each link length and radius to be 1/2 %. They could be independent, but we saw no obvious reason to set them differently. The mass and inertial moments of the links are recomputed after each growth step. The inertial tensor is also recomputed after each growth step.

In our experiments we have compared 2 rates of growth: (1) one where a growth increment of 1/2% occurs every 128 practise moves, and (2) one where the same growth increment occurs every 50 practise moves. In the first case of a step increase every 128 movements, the total growth during the 2k movements produces a mass increase of each link of about 1.27 times the original mass, with a length and radius increase of 1.083. In the second case of step increases every 50 movements, the mass of the limb is increased 1.81 times, with total increases in link lengths and radii of 1.22. The slow growth rate was arbitrarily selected, but the faster grow rate of 1/2% increment in dimensions every 50 practise moves was found to be near the growth limit where some amount of improvement due to learning still occurs.
Figure 5.5: A comparison of learning performance between movement representations using equilibrium positions (top plot) and movement representations using torque step functions (bottom plot) during growth spurts. The growth spurts occurred every 128 movements. Compare these plots with the corresponding no-growth plots of Figure 5.4. Data for each plot was from 10 runs, with 2000 movements per run.
Figure 5.5a: The worst, median and best trajectories after 2k practise trials each. During the practise trials, each link of the arm would increase in radius and length by 1/2 percent every 128 trials. These trajectories are from the same data set that produced the top plot of figure 5.5. The top trajectory has a relative accumulated jerk value of 304.0. The middle trajectory has a relative accumulated jerk value of 5.93. The bottom trajectory has a relative accumulated jerk of 0.663. These trajectories were produced from schemas with 8-10 equilibrium points each.
Figure 5.5b: Three trajectories from the data set that produced the bottom plot of figure 5.5. The top trajectory has an accumulated jerk of 39.6, the middle trajectory has an accumulated jerk of 7.16, and the bottom has an accumulated jerk of 1.17. As with figure 5.5a, the arm was growing in 1/2 percent jumps every 128 practice moves. Notice that the upper trajectory overshoots its final position and doubles back. There is a hint of this in the middle trajectory as well.
5.2.2 Interactions between growth and learning

The results of the first, slower growth rate are presented in Figure 5.5. Figure 5.5a and Figure 5.5b contain sample trajectories taken at the end of 2000 practice attempts. As with figure 5.4, learning with equilibrium position control is presented in the upper graph of figure 5.5, and learning with torque step functions is presented in the lower graph of figure 5.5. Similarly, ten runs of 2000 practice movements per run contributed to form the upper graph, and an additional ten runs contributed to the lower graph. There are several observations we would like to make.

The first observation has to do with the effect of the growth steps on the learning curve for individual schemas. In order to show how the total accumulated jerk of a schema is affected by a growth step, we have included as Figure 5.5c the learning curve for one schema instance during a run of 2000 movements. This learning curve plots progress in one cost functional only, the accumulated jerk. In the figure, growth steps often produce the large vertical step increases in the accumulated jerk value. Figure 5.5d shows a typical learning curve without growth for comparison. The equilibrium point control sequence for the previously shorter and less massive arm often produces a movement with suddenly higher jerk, when applied to the arm after a growth step. Thus, many of the step increases in jerk in the figure result from movements made immediately after growth steps.

**Figure 5.5c:** Learning progress for one schema during its run of 2000 movements. Growth steps of 1/2% in length and radius occurred every 128 steps. Only movement attempts that are taken as progress toward minimizing jerk are plotted. The exploratory (search) movements that did not produce progress were omitted.
Figure 5.5d: A typical learning curve for one schema using equilibrium position control, but without growth. Compare with figure 5.5c.

For the second observation we will refer to figure 5.5. Though the information is not presented in the figure the search task is significantly different for the datasets contributing to the upper graph of figure 5.5 vs. the datasets contributing lower graph (i.e. for equilibrium point control vs. torque control). For example, the number of equilibrium positions per movement increases from 1 to about 8 by the 1000th movement in nearly all of the runs contributing to the upper graph. On the other hand, the runs contributing to the lower graph in figure 5.5 typically reach only 4 to 6 (sometimes 8) CSE's per schema by the 2000th movement. This difference is due to the nature of the practise strategy for the task. The practise strategy increases the number of control structure elements every time the learning process slows down below a certain amount. Since the overall learning rate for equilibrium control is higher than with torque control, the learning plateau is reached earlier for equilibrium control. Hence, the number of CSEs in any schema instance increases at a higher rate for equilibrium control.

To put this another way, the rate at which the temporal mesh is refined in the lower graph is roughly half that of the upper graph. The result of proceeding faster is that equilibrium position schemas increase the dimensionality of their action space much faster than torque step schemas. By the 1000th movement the dimensionality of the action space in the upper curve is $8 \times 2 = 16$, while the dimensionality of the action space in the lower curve is $4 \times 2 = 8$. The dimensionality for equilibrium position control means that much more search will be required to decrease the cost functionals further.

When the dimensionality of the action space gets high enough, the time to search the action space for an improvement approaches or exceeds the time between growth steps. As mentioned earlier, growth steps often produce cost functional step increases. When this happens, the cost setbacks due to the growth steps can occur faster than the search process can restore the schema to a lower cost.

---

6 The reason for the difference, in more detail, is that under equilibrium position control the system makes faster progress in minimizing its cost functionals. When it bottoms out at the best minimum it can find, it tries to get itself off this minimum by applying a variation, such as splitting an equilibrium position into two equilibrium positions (thus refining the temporal mesh further) and searching the newly expanded action space.
Figure 5.6: A comparison of learning performance between movement representations using equilibrium positions (top plot) and movement representations using torque step functions (bottom plot). The dashed lines plot learning without growth of the arm. This is the same data as in Figure 5.4 (the median traces). The solid lines plot learning while the arm is growing. This is the same data as in Figure 5.5 (the median traces). Notice that the first half of the upper pair of plots and the entire lower pair of plots show essentially no difference in learning performance when the arm is growing as opposed to when its not.
Figure 5.7: A comparison of learning performance between movement representations using equilibrium positions (top plot) and movement representations using torque step functions (bottom plot) during fast growth. The growth steps occurred every 50 movements. Compare these plots with the corresponding slower growth plots of figures 5.5 and 5.6. Data for each plot was from 10 runs, with 2000 movements per run. The parameters that were changed to accommodate the fast growth were the controller gains for the equilibrium position controller, which were tripled, and the torque limits for the torque step functions, which were tripled.
Figure 5.7a: Some trajectories from the high stiffness (3x controller gain), equilibrium position controlled data of figure 5.7. The accumulated jerk of these trajectories is in the range of 5-20 in the same relative units of the other figures. These examples of less than perfect learning progress share several characteristics of infant reaching movements. Their indirect path to the target is certainly reminiscent of infant reaching movements, but more importantly, close examination of these and similar cases show that the inflection points (regions of high curvature) of the trajectories are also locations where the velocity of the hand is at a local minimum.
Figure 5.8: A comparison of learning performance with the movement representations using equilibrium positions at two different controller gains both during fast growth. The growth steps occurred every 50 movements. In the top graph, the solid plot with "error" bars corresponds to data with controller gains set to 3x (relative to 1x in figure 5.6 top), and the dotted plot used controller gains set to 2x. In the bottom graph, the solid plot with error bars corresponds to 2x controller gains, and dotted used 3x.
Consequently, the rate at which the search process can compensate for the changes produced by the growth steps becomes slower. Due to the higher dimensionality, this problem is worse for the upper graph in relation to the lower graph. This is the reason for the increase in accumulated jerk in the upper graph roughly between movements 1000 and 1800. In other words, at 8 CSE's per schema (from movement 1000 to 2000 in the upper graph), the search process can barely keep up with the growth. The graph wanders up and down because the stochastic search sometimes does well against the growth and sometimes does not. At 4 CSE's per schema (the lower graph of figure 5.5), the search process can still make progress against the growth. However, by movement 2000 the equilibrium controlled schemas have nevertheless improved considerably, and match or exceed the performance of the runs in the lower graph. To compare the trajectories generated by equilibrium position control vs. torque step control compare Figure 5.5a with 5.5b.

Figure 5.6 clarifies this last point by overlaying figure 5.5 with figure 5.4. Thus, the growth vs. no growth graphs can be compared. In the upper graph, for the first 1000 movements or so, the growth and no growth learning curves are similar. However, between movements 1000 and 2000, the difference is dramatic. The learning curve for the growth case actually gets worse for a while, for the reasons discussed above. The differences between the growth and no growth experiments can be used to illuminate several aspects of infant reaching movements.

First, it is apparent that the rate of growth must be balanced by the rate of cost functional reduction if learning is to progress. Furthermore, this balance is directly affected by the dimensionality of the search space. Thus, for learning to make progress, the dimensionality of the search space must be constrained inversely by the rate of growth. This may be the reason for the phenomenon of "movement units" in infant reaching movements. It may be that during periods in which the arm is growing quickly, clusters of equilibrium points are modulated as a unit (e.g. add the same vector to each), rather than having all the equilibrium points modulated independently. This clustering reduces the dimensionality of the portion of action space that is searched as the learning system tries to keep the movement on target. Our simulations show that this dimensionality can be low (e.g. 4-10 equilibrium point clusters) and still support accurate, low impact reaching movements. Higher dimensionality is needed for smoother movements. Thus, when the growth of the limbs slows down, the dimensionality of the action space can be allowed to increase. The consequence of this increase in dimensionality is that the accumulated jerk cost functional can be reduced to much lower values as the clustering of equilibrium points can be eliminated. This elimination of movement unit clustering would appear as if the 4 or more equilibrium point clusters eventually became 1 cluster, i.e. 1 movement unit. This is what happens developmentally.

Second, the increase in accumulated jerk that is due to an imbalance between the dimensionality of articulator search and the rate of growth can be mitigated by increasing controller gains. Figure 5.8 shows additional data we have collected concerning equilibrium controller gains. The figure shows that increasing the gains of the equilibrium position (feedback) controller can restore learning progress for trajectories using 8 (or perhaps a few more) equilibrium points. This would lower the right half of the upper solid plot in figure 5.6. That is, increasing stiffness of the arm restores the rate of learning
during growth. This cannot be pushed too far, of course, but moderate increases in gain, such as 2x or 3x, can substantially improve learning performance.

Third, the conditions of fast growth combined with high stiffness produce movements, in the early stages of learning, that have a much different visual gestalt than the movements that are encountered with either lower stiffness or with the alternative torque step functions. These intermediate products of the learning process tend to take a more twisted route to the target as they are pulled more powerfully to the equilibrium points in the sequence (see figure 5.7a).

Fourth, the more twisted trajectories, that occur under conditions of fast growth and high stiffness of the arm, evidence the same curvature-speed relationship that was discussed in chapter 2. This outcome of our simulations will be discussed in more detail in a later section of this chapter.

5.2.3 Section summary

Our findings regarding learning to reach while the arm is growing can be organized around the following points. First, we have found that equilibrium position control is more conducive to learning arm movements than direct (not feedback controlled) torque control. This is somewhat true for learning without growth, but more dramatically true when growth of the arm is occurring concurrently with learning to reach. Second, since the growth process is continually changing the length and mass of the limbs, growth has a very destabilizing effect on the ability to make accurate and smooth reaching movements. For learning to progress adequately under such conditions two changes to the arm/learning system are sufficient. One change is to cluster the equilibrium positions into a small number of movement units. Another change involves increasing the stiffness of the arm/control system to compensate somewhat for the increased mass of the arm. The results of these changes not only support the development of effective reaching movements during relatively fast growth, but produce movements that share important characteristics with infant reaching movements. One such characteristic is an indirect, nonlinear (e.g. somewhat twisted) path to the target during the early development of reaching. A second characteristic is the emergence of the same speed-curvature relationship within these nonlinear paths as is present in infant arm movements.

Thus, we have found developmental reasons for the equilibrium point hypothesis, the existence of movement units, and the high stiffness of the arm system in infant reaching. As we will discuss in a later section, our simulations can also identify how and under what circumstances arm movements with a human-like speed-curvature relationship develop.
Figure 5.9: Comparative examples of trajectories using either a minimum jerk functional or a minimum time functional, instead. The upper two examples are the least linear (top) and most linear (middle) from a set of 10 runs using the minimum jerk functional. The bottom example is a higher linearity example selected from a set of 10 runs using a minimum time functional. As in many of the previous figures, each run corresponds to a sequence of 2000 practise trials for an individual schema.
5.3 Experiment 3: Comparison of minimum-time and minimum-jerk constraints

In chapter 2, we suggested the possibility that differences in some of the findings of apparently similar experiments could be accounted for by differences in subjective interpretations of the task. These differences in interpretation could result in different cost functionals being applied to different tasks, even though each task has the same goal (e.g. reaching for and touching a target). Since the cost functionals are employed as measures of learning progress, very different adaptations should result from what on the surface would appear to be the same task.

In particular, we proposed to compare two versions of the reaching task. In one version, the total accumulated jerk cost functional (see chapter 3 for definition) would be used, and along with it the usual costs for final position error and final velocity error ($V_f=0$). In addition, we considered the possibility of using a slightly higher stiffness than usual, as well as a small amount of friction. This choice of functionals and parameters was to mimic a more careful and deliberate reach attempt, where the infant would be trying to grasp the target at the end of the movement, rather than simply knock it down. However, since the results of the previous section have already shown that higher stiffness will produce less linear trajectories early in the learning process, we chose not to increase the stiffness in this experiment (i.e. make the comparison with the same stiffness).

In the second version of the reaching task, we replaced the minimum-jerk constraint with a minimum-time constraint. That is, we replaced the accumulated jerk cost functional with an accumulated time cost functional (e.g. simply measure the time to target and use the value as a cost). The final position error cost was retained, but the cost for a non-zero final velocity was eliminated. Friction was set to zero.

As in the experiments discussed in chapter 2, we used the quotient of the arc length travelled by the hand with the euclidean distance between the starting and ending points of the movement, as our measure of linearity. With this measure a perfectly straight trajectory will have a linearity of 1.0. As the trajectory is less linear, this measure will assume increasing values above 1.0.

The results of these experiments are shown in Figures 5.9 and 5.9a. First, consider Figure 5.9a, which shows the linearity values for the minimum-time experiment on the left, and the minimum-jerk experiment on the right. These linearities fall into two almost disjoint populations, with the boundary at linearity = 1.04. Clearly, the minimum-time experiment produced much more linear trajectories. It should be mentioned that these results might not be the same if the runs had exceeded 2000 practise trials per run. With more practise, the minimum-jerk experiment will show progressively more linear trajectories. However, as we have mentioned earlier, the axial component of jerk must be reduced a great deal before the transverse component will show noticeable reduction. Thus, the findings presented in Figure 5.9a are probably only relevant to infant studies within about the first year of development.

Figure 5.9 shows some of the trajectories that correspond to these results. The upper two trajectories are the least linear and most linear trajectories from the minimum-jerk experiment. The lower trajectory is the most linear trajectory from the minimum-time experiment. Although, all the minimum-time trajectories look similar.
Figure 5.9a: (Left) Linearities of a set of runs using a minimum time functional instead of a minimum jerk functional. (Right) Linearities of a set of runs using a minimum jerk functional and a small amount of friction. Equilibrium position controller gains were set at 1x in both cases.

The tentative conclusion that we would draw for developmental studies of eye-hand coordination is that it is indeed possible for very subtle differences in the way infants perceive a task to change their strategy or approach to the task, and hence the outcome of the experiment. At least in the case of reach linearity, it is possible that the difference between reaching to grasp an object and reaching to contact (or knock) an object can produce markedly different outcomes.

5.4 Experiment 4: Speed-curvevature relationship

A fortuitous discovery occurred when we were examining some of the intermediate trajectories produced during the fast growth, high stiffness learning experiments (recall figures 5.7 and 5.7a). The trajectories, that we would normally consider to be disappointing, exhibited a phenomenon absent from straighter trajectories. The phenomenon, of course, was a relationship between the speed and the curvature of the trajectory at inflection points (see Figure 5.7a, 5.10a, and 5.10b for examples of such trajectories). As discussed in chapter 2, infant and, under special circumstances, adult reaching movements show a speed-curvature relationship where high curvature regions of reaching trajectories are accompanied by local speed minima.

What was surprising about finding the same speed-curvature relationship in our data, as that found in infant movements, is that it emerged naturally from the combination of equilibrium position control, fast growth, high stiffness, and accumulated jerk cost minimization in the context of our basic learning algorithm. In addition, the location of the local speed minimum is a very accurate predictor of the location of the local curvature maximum under the conditions just listed. By re-examining our data we have found a very strong dependence on this combination of factors.

Figure 5.11 shows a summary of this re-examination. We have run about 170 data runs during code debugging and the subsequent preparation of this chapter. Each run consisted of 2000 practise trials for one schema, with one starting and ending position. Of these 170 runs, 71 did not involve growth and the remaining 99 did. In the 71 no-growth cases, we did not see speed minima accompanying local curvature maxima. In fact, we did
not see very many instances of high curvature in the trajectories, especially as compared with their occurrence in the growth data. However, we cannot conclude that growth is necessary for the occurrence of a speed-curvature relationship, since there are other ways of producing trajectories with high curvature regions. Furthermore, all of the no-growth data involved a low stiffness setting for the arm. That is, our data sample has not properly separated the factors of growth and stiffness, so we cannot confidently say whether one or the other or both are necessary. We can say that both are sufficient, assuming that the trajectories are controlled as sequences of equilibrium positions.

The table of Figure 5.11 was produced by examining trajectory-related graphs identical with those appearing in Figure 5.10a,b,c. With such graphs it is very easy to spot speed minima. However, identifying and locating curvature maxima is much less reliable by inspection. We plan to automate this analysis with the obvious computer algorithms in the near future. Our visual inspection did reveal that most of the cases where the speed-curvature relationship does appear are cases where rather steep inflections have occurred. From the more detailed data of the 2000 trial sequences, it appears that when the inflection point has remained while the accumulated jerk functional has continued to be minimized, a surprisingly low-jerk movement results. Such movements show the speed-curvature relationship, where the speed minima occur at exactly the same locations in the trajectory as the curvature maxima. Figures 5.10a and 5.10b are typical examples of this. To show that a speed minimum need not co-occur with a curvature maximum, we have included Figure 5.10c. Figure 5.10c represents a slow growth and low stiffness case. Notice that the graphs of Speed(t) vs. X(t) and Y(t) vs. Speed(t) show much less variation than the corresponding graphs in Figure 5.10b or 5.10a.

The greater activity in speed(t) hints that a combination of slightly awkwardly placed equilibrium points, plus a higher energy of attraction to these points, might be producing the inflections in the trajectory. According to our data, the inflection points tend to find their way into the movements primarily because of the combination of fast growth and high stiffness, though other factors could produce inflections. When an equilibrium point sequence has been adapted to a given set of initial and final conditions, such that the functionals corresponding to these conditions are reasonably close to their minima, and if the jerk cost functional has been reduced somewhat, then a growth step will often produce a step increase in jerk and will warp the trajectory considerably. The high stiffness in combination with the now inappropriate locations of the equilibrium points tend to produce bends in the trajectory. In the aftermath of the growth step, the learning algorithm pursues minima with respect to the cost functionals. As the accumulated jerk is lowered, the warped curve may "settle" into a curve that still has the inflection point, but which may be near a local minimum in the accumulated jerk. At this stage of events the trajectory will show the speed-curvature relationship. Continued iterations of the learning algorithm would eventually remove the inflection. However, if growth continues at a rapid pace, new inflections will appear and disappear.

Admittedly, this discussion has been speculative, with a very informal analysis of the data. However, we have offered our viewpoint to indicate how we would explore the processes that contribute to the emergence of a speed-curvature relationship.
Figure 5.10a: The upper left graph shows the trajectory of the tip of the 2-d simulated robot arm. This trajectory developed during practice trials in which a growth step of 1/2% in length and radius occurred every 50 movements. The equilibrium point controller gains were set high at 3x. Notice that the trajectory graph contains 3 clear local curvature peaks (inflection points), one of which is indicated with an arrow. Compare the X locations of these inflection points with the 3 local speed minima on the lower, narrow plot of Speed(t) vs. X(t). Compare the Y locations of the inflection points with the local speed minima on the upper right plot of Speed(t) vs. Y(t). Notice how the inflection points can be precisely located by these minima (e.g. one is at (405, 385), the one indicated with an arrow is at (370,325)). This example is actually typical of fast growth, high stiffness trajectories during intermediate stages of learning (eventually they loose the inflections and become straight).
Figure 5.10b: As with Figures 5.10a and 5.10c, the graph at the upper left displays the trajectory of the tip of the 2-d simulated robot arm. The growth rate and stiffness are the same as for Figure 5.10a. This trajectory is parameterized by time as is the speed of the tip of the arm, denoted Speed(t). The speed appears in the narrow right and bottom graphs against X(t) and Y(t), respectively. This arrangement allows direct location of speed minima on the narrow graphs so that the corresponding (X,Y) location can be found on the trajectory. For example, on the bottom Speed vs X graph, there are two local minima in speed, one at X=370, and at X=410. On the Y vs Speed graph, there are also two local minima in speed, one at Y=361, and at Y=395 (N.B. Y=395 is not an endpoint, but a dip, though this is not obvious from the plot). Projecting these minima onto the X(t) vs Y(t) plot produces two points, (370, 395) and (410, 361), at which the speed of the tip was at local minima. By inspection these points are at or very near the local curvature maxima, which are indicated by the small arrows.
Figure 5.10c: An example of a local curvature peak that is not accompanied by a speed minimum. The equilibrium point controller gains were set to 1x, so this is not a high stiffness example. The rate of growth was slow at 1/2% length and radius increase every 128 movements. Notice that there is no dip in either the Speed(t) vs. X(t) plot, nor in the Y(t) vs. Speed(t) plot. Consequently, this example does not exhibit the speed-curvature relationship discussed in the text. This case is typical of other trajectories at the same rate of growth and low stiffness. Only one trajectory in about 40 at this rate of growth and stiffness showed a local curvature peak that could be located with a clear speed minimum. In this example there are no minima aside from the end points. Note that the shape of this trajectory is exaggerated by the magnified Y(t) scale.
Case description

Growth

Equilibrium position control

slow growth:
1x gain, speed minimum at high curvature 01
1x gain, high curvature without speed minima present 02
1x gain, remaining cases 08
2x gain - (not analysed) 10

fast growth:
1x gain - (not analysed) 10
2x gain - sharp curvature peaks, accurately located by speed minima 15/10
3x gain - sharp curvature peaks found in data, accurately located by speed minima 17/10

Direct torque function control

slow growth:
target overshoot,
where arm stopped and reversed near target 02
(i.e. infinite curvature, speed = 0)
remaining cases 08 10

fast growth:
high curvature, but speed minima lagged in time,
occuring in nearby region of low curvature. 02
sharp curvature peak with no minimum in speed 02
remaining cases 14 10

No growth

Equilibrium position control 37 00
Direct torque function control 34 00

Legend:

ns/c = no speed/curvature relationship present
s/c = speed/curvature relationship found
na = not analysed
d1/d2 = d1 high curvature locations found in d2 runs

Figure 5.11: Informal summary of the occurrence of a speed-curvature relationship in our data. These data come from a quick visual inspection of trajectory and speed graphs, identical to those of Figures 5.10a,b,c. This summary is not meant as a substitute for a more careful analysis of the conditions under which a speed/curvature relationship is found. Such an analysis is planned for a future study. However, the patterns in the data are striking and suggest that further consideration is warranted.
5.5 Experiment 5: The basic visual grasp task
5.5.1 Learning to make saccades to stationary targets.
5.5.1.1 Brief description of hardware and software simulation.

In chapter 3 we described a visual grasp learning task, and provided a mathematical description of the task. We also described, though briefly, the MIT vision machine, which we have used to demonstrate the results of the learning process. In this section, we will describe the learning task in more detail and present the results. Because the results we will present required relatively large numbers of trials, it would not have been practical to run the entire learning process on the vision machine hardware. Instead, we sped up the learning process by using a computer simulation of the hardware that duplicates its function reasonably well. In fact, we can replace the simulated senses and motors with the actual hardware and obtain accurate eye-head movements from the same GRBF network. We should mention that the training can be done with the vision machine hardware, but we chose not to go this route beyond testing to make sure that it worked. 7

For the simulated head-eye, we used 2 degrees of freedom: horizontal movement from -45 degrees to +45 degrees, and vertical movement within the same range. As with the hardware, no form of visual recognition was done with the simulation. Only one target was in the field of view at a time. Also similar to the hardware, the only form of control was to specify the next position as pair of angles, (pan, tilt), and wait for the controller to return the amount of time the movement took, and the resulting resting position of the head-eye. 8 The hardware and controller are not set up for smooth pursuit. We also chose to restrict our attention to saccade-like 9 head-eye movements with the simulator, and did not experiment with velocity control. Within the field of view of the simulated head-eye, targets with specified location and velocities would appear either randomly, or according to a program.

---

7 We also need to mention that we used the simplest possible substitute for visual recognition with the hardware. For moving targets, we isolated patches of movement by taking differences of successive frames. In the resulting differenced image, anything stationary becomes 0, except for noise. The noise can easily be removed with thresholding or smoothing, or a combination. A simple algorithm computed the smallest rectangular enclosure around each such patch. Any number of simple heuristics can be used to pick a particular patch and find it again once the eye-head has moved. That is, the heuristics remain simple assuming that there will be few such moving patches, and that they will be very distinct in size or shape. For stationary targets, we used very contrasty targets (black on a white background) and a simple combination of edge detection (Canny, ref.???) edge contour following, and a least squares fit to find continuous, straight edges in the contour (Drumheller, ref.??). Again, simple heuristics allowed us to find distinctive corners in high-contrast 2-d scenes. For example, if the system picks the corner with the most acute angle, or the longest edges, and if the scene is very simple, then it will be able to find this corner after the eye-head system moves. Parallel versions of these algorithms were written to run on a Connection Machine.

8 With the head-eye hardware there are times when the commanded position and the resulting position are not equal. There is an accumulated position error that can be noticeable after a dozen or so movements.

9 We refer to these movements as saccade-like because they are single jumps from one resting position to another.
5.5.1.2 Learning modules for the stationary target task.

We separated the visual grasp task into two versions: visual grasp to stationary targets and visual grasp to moving targets. This is an artificial separation, but as it turns out, the moving target version of the task is much more quickly learned after the stationary task has been mastered.

For stationary targets, the task is to learn to make accurate saccades as targets appear in the field of view. The cost functionals associated with the task include a position error cost. This cost is not assessed unless the target falls within a perifoveal region whose size is 10% of the field of view. If the field of view is mapped to a 1000 pixel square region, then a concentric 100x100 pixels square is the region within which position error is returned as a cost. If the target falls outside the central, 100 pixel square region after the eye-head movement, no position error is provided. This is an arbitrary choice, but we picked it because it is conservative. Surely the learning task becomes much easier when this region is larger. The cost for final position error is simply the euclidean distance between target position and the center of the field of view (see expression 3.3.1).

There is another choice that we could have made to make the learning task easier. We could have kept the x and y dimensions separate, learning horizontal control separately from vertical control, with separate horizontal and vertical position error dimensions associated their respective control dimensions. This too is an arbitrary choice. However, we would like to explore mechanisms by which such separations of error dimensions and subsequent associations with action dimensions can be discovered. So it seemed that our unseparated version of the task might be a good place to start experimenting.

A second cost involves the length of time between the beginning of the first saccade-like movement in a sequence and the end of the last such movement. How many saccade-like movements comprise one sequence (the sequence of actions of one schema by our definition) can vary. Expression 3.3.2 in chapter 3 gives a definition of this cost. No other costs are needed for the stationary target task.

This second cost must be applied carefully since it does not in itself distinguish between few saccades or many if they add up to the same amount of time. For this reason we added an constant offset to the time associated with each saccade-like movement. An example of this is shown in figure 5.12b. Saccade time, and therefore cost, is proportional to delta-angle plus a small added cost for each saccade. Figure 5.12a shows a quadratic term added to the linear and constant terms for saccade time vs. angle. This quadratic term is a very approximate account of a non-linearity in the head-eye hardware for large angles. We have not done a careful fit to head-eye data, but we have captured the effect qualitatively.
Figure 5.12a,b: Two functions of saccade duration vs. saccade angle used for the head-eye simulation. The left graph (5.12a) shows saccade duration vs. angle for the case where large angle saccades take a long time because the controller "hunts" for the equilibrium position. The right graph (5.12b) shows a linear saccade duration vs. angle function. These graphs only capture performance of the head-eye hardware qualitatively. The actual saccade durations for the hardware are much longer.

5.5.1.3 Results of experiments with the stationary target task.

The remaining details of the stationary target simulations concern the initial conditions and the complement of variations used as a practise strategy. First, the system starts with an empty topographic map. That is, the GRBF network for the task has no data points in it (an empty network). As targets appear, the system must retrieve an appropriate schema, or create a new one. After schema creation, the first variation operator to fire picks the initial number of CSE’s for the schema, and assigns their initial content. For the stationary task, the variation picks a random number between 1 and 12 for the number of CSE’s, and randomly assigns action values to each CSE in the schema. The context for the schema consists of the coordinates of the target on the retina, and articulator coordinates of the head-eye system at the time the schema is requested. Figure 5.13A shows such an initial head-eye trajectory. The crosses in the figure indicate gaze locations of the head-eye, with increasing cross size indicating succession in time. The circle to the lower left of the frame is the target. The initial gaze location is indicated by the cross in the center of the frame.

Clearly, the first trajectory is unimpressive both with respect to final position accuracy, and certainly time-to-target. It is also clear that mutation operations, which will vary the action values of each saccade in the sequence, will not be sufficient in minimizing the time-to-target cost functional. Mutation operations can improve the final position error, but cannot shorten the saccade sequence.

For the latter purpose, there are 2 variation operators that are constituents of a practise strategy for this task. One of these is a combine operation (refer to chapter 4, figure 4.2). The combine operation for this task is slightly different from the combine operation discussed in chapter 4. Recall that the combine operation takes two adjacent CSE’s and replaces them with 1 CSE whose duration is equal to the sum of the 2 durations. In this case, the duration of the CSE is not commanded to the controller, but returned by the controller. Consequently, the combine operation only needs to generate
an action vector, in this case a pair of equilibrium positions, (pan, tilt), for the head-eye. The average of the pans and tilts of the previous 2 CSE's works fine. So the next questions are when and how does the system pick CSE's to combine.

Figure 5.13B contains a hint for a very simple condition for the variation operator. In addition to the crosses for eye positions and a circle for the target position, there are 2 ovals, each surrounding a pair of crosses. If we were to fast forward through the simulation, based only on mutation operations, but continuing from frame B, we would see that the crosses within an oval would move toward each other. Eventually, these surrounded crosses would end up with the same (pan, tilt) action values. This happens because of the minimum-time functional. Even without the combine operation, mutation operations and the minimum-time functional tend to push extraneous equilibrium positions together. Without a combine operation, the simulation would run until all the crosses were bunched up in either of 2 places, the beginning of the sequence or the end. Thus, all we need as a condition for the combine operation is that the actions of two adjacent CSE's are componentwise almost equal.\(^{10}\) As it turns out, simulations can progress much more rapidly if the combine operations is significantly more aggressive. That is, it should periodically and stochastically try to combine adjacent CSE's within some threshold radius of each other. If the combine operation fails to improve the cost functionals, it can be undone.

Recall the two equilibrium position controller functions of Figure 5.12a and b. The controller corresponding to figure 5.12a produces very long saccade durations for large angles. It is in fact the case that a shorter time-to-target is accomplished with several saccades, if the saccade distances are within the linear range of figure 5.12a. Such a result is presented in Figure 5.13, frame C. For this frame, 127 practise trials were run from the same initial conditions as produced the frame above it, but with the non-linear controller of Figure5.12a.\(^{11}\) On the other hand, frame C' was produced from a continuation of the trials that produced frames A and B, but in this case the linear controller of figure 5.12b was used. Typically around 100 practise trials are needed to reach the final, 1 saccade solution with 1% position accuracy.

Finally, there is the second variation operator alluded to several paragraphs back. It is possible for the system to combine CSE's down to 1 remaining CSE, which is a local but not a global minimum. If the non-linear controller is being used, then there's no obvious way for the system to get to a better minimum, because this requires more than 1 CSE in the schema. For this reason, a split operation is also included in the practise strategy for this task. Again, this split is slightly different from the split used for the arm task. Instead of splitting a CSE into 2 CSE's with the same action vector, but halving the duration for each (see figure 4.2), the new split operates on the action vector and leaves

\(^{10}\) An additional combine-like operation is also useful. Often the repeated application of mutations (stochastic search) produce sequences that reach the target early, before all of the saccades have been executed. The additional operation, just drops CSEs off the end of a schema when they haven't been executed for many trials.

\(^{11}\) Note that the minimum-time constraint for this version of the task does not have a unique solution. Many points along the line between the initial and final gaze locations will produce the same cost. In such cases the learning algorithm will wander around the minimum surface as long as it is left running.
Figure 5.13: An assortment of simulated eye-head trajectories toward a stationary target. The circle in each frame is the target. The crosses indicate the center of the field of view of the eye-head system. A sequence of saccade-like eye-head movements in time is indicated by successively larger crosses connected by textured lines. The smallest cross in the center is where the eye-head system initially directs its gaze. The target cross indicates the final position of the eye-head for the schema. It takes ~100 trials to go from A to C or C'.

This first attempt is a random sequence of 10 equilibrium points

After further progress several equilibrium points have been combined away

With a non-linear equilibrium position controller, two equilibrium pts. give best time-to-target

With a linear equilibrium position controller, the best time-to-target is 1 pt.
the duration unchanged. If the action vector of the CSE to be split is <pan_1, tilt_1>, and if the context of the CSE (i.e. the previous head-eye location) is <pan_0, tilt_0>, then an extra CSE is interposed between these two locations at <(pan_0+pan_1)/2, (tilt_0+tilt_1)/2>. With both the combine and split operations, it is possible for the system to learn to make efficient saccade-like movements. Figure 5.13C shows a 2-saccade movement that was learned with the non-linear equilibrium position controller. Figure 5.13C’ shows a 1-saccade movement that was learned with the linear equilibrium position controller. Each of these saccade sequences are about optimal for the respective task conditions.

5.5.2 Learning to make saccades to moving targets.

5.5.2.1 Brief description of hardware and software simulation.

There are some additional observations to be made about the hardware concerning its use for this more demanding task. Saccade-like head-eye movements must be able to exceed the angular velocity of the target with respect to the head-eye location. The difficulty in tracking a moving target with a more massive structure than the eye can be experienced rather easily by keeping your eyes fixed (as much as you can do this) and trying to track objects with head movements only. For this reason, as well as those discussed previously, it was important to study the moving target version of the head-eye task primarily in simulation.

With respect to the simulations, the additions needed for this task included simulating targets moving across the field of view, as well as moving projections of the targets through optics onto a simulated retina. During simulations, targets with specified location and velocities would appear either randomly, or according to a program. Simulated target motion was always linear, and constant velocity.

5.5.2.2 Learning modules for the moving target task.

There are three significant differences that were introduced into the moving target task that were not present in the stationary target task. The first difference is in the sequence of events that constitute the task, which now includes a waiting period after a saccade is made for observing the target's trajectory. A timing diagram of the task is presented Figure 5.18, and will be discussed below. The second difference is related to the first, and it involves an additional cost functional that is included for minimization with the others (see chapter 3, expressions 3.3.1, 3.3.2, and 3.3.3). This cost functional was defined in chapter 3 by expression 3.3.3. The third difference is that the moving target task presupposes that the stationary task has been learned, at least partly. This difference is not necessary, but it allowed us to understand how a primitive "anticipation" of a moving target can be learned without planning. One additional minor difference between the stationary target task and this one is that the number of CSE's per schema (hence the number of equilibrium points for a movement) is usually less than for stationary targets, since the target, when moving moderately fast, will be out of view after 3 or 4 saccades.

---

12 The task is more difficult for the head-eye hardware which is larger and quite massive being constructed of aluminum, and having several stepper motors riding the swing arm with the cameras (Figure 3.2).
(1) Acquire image;

(2) select target;

(3) retrieve or otherwise generate a movement;

(4) execute the movement (mostly flight time, and settle time);

(5) acquire another image;

(6) process image to determine whether target is in perifovea region;

(7) if yes, determine direction of target movement relative to fovea;

(a) if target is approaching fovea center, continue tracking;

(b) if target is receding from fovea center, discontinue tracking;

(c) if target is not approaching or receding, discontinue tracking;

(8) compute functionals:

(a) cost for final position error (expression 3.3.1);

(b) cost for saccade sequence time (expression 3.3.2);

(c) cost for time until target is nearest fovea center (expression 3.3.3).

Figure 5.18: The timing sequence for the head-eye simulation (or hardware) when learning to make saccades or saccade sequences to moving targets.
The timing sequence for the moving target task is shown in Figure 5.18. Steps 1 through 6 are the same as for the stationary target task (or the arm movement simulation). Step 7 and the cost functional of step 8c are new for this version of the task. Note that if there is more than one CSE in the schema, steps 3 through 7 are repeated for each CSE. There is another loop, which is composed of steps 5 through 7. This loop is inside the larger loop of steps 3 through 7. The added cost functional, 8c, is computed from the results of the inner 5 through 7 loop. These steps repeatedly sample the target position with the head-eye remaining stationary.

The idea for the extra steps and functional is fairly simple. After a saccade is made and the head-eye settles, it checks to see where the target is and where it's headed in relation to the center of the fovea. If the target is very nearby and approaching the fovea center, it is tracked to determine when it passes closest to the fovea center, and what the closest distance is. After the target passes its closest point to the center of the fovea, and the target is receding from the fovea region, tracking is discontinued and the distance and time of the point nearest the fovea center are recorded. The distance and time of closest proximity to the fovea center are then used by those cost functionals that determine the final position error and the total time of the head-eye movement. These cost functionals favor head-eye landing positions that are ahead of the approaching target, rather than landing behind it. Thus, if a mutation operation moves the head-eye behind the target, but closer to the target than the previous practise attempt, the mutation will be rejected if it would be replacing a previous saccade that landed ahead of the target but in a position where the target subsequently passes over the fovea center. This slightly biases the learning process so that the target is more likely to pass over the fovea after the saccade is made, even if it does so when the head-eye is stationary and waiting.

5.5.2.3 Results of experiments with the moving target task.

The simulations that we are about to discuss have, as part of their startup conditions, a GRBF network that has practised with stationary targets for the regions of articulator space that will be needed for the moving target experiments. New schemas for tracking moving targets will usually have 1 to 4 CSE's, the count determined randomly, as with the stationary target experiments. However, unlike the stationary target experiments, these initial CSE's are not given action vectors randomly. Instead, the nearest stationary target data will be retrieved from the GRBF network and used to fill these CSE's. For example, if a moving target is first spotted at location (300.0, 200.0) with velocity vector (245.0, 5.0), the nearest existing or interpolated data will provide an action vector for position (300.0, 200.0) with velocity vector (0.0,0.0). This action vector will, of course, move the head-eye gaze so that it centers on position (300.0, 200.0). Meanwhile, the target will have moved. While this is a miss, it is not nearly as bad as a random first action. In fact, we will shortly see that with a very simple practise strategy, a sequence of such near misses can be very quickly turned into a good schema for saccade-like tracking.
Figure 5.14: Learning progress for low speed targets (250 pixels/sec.), and for high speed targets (700 pixels/sec.) are shown in frames 1a,b and 2a,b, respectively. The cases in each of these datasets started with 2 or 3 CSE’s per schema. No practise strategy was employed for dataset A or dataset B. Frame 3 shows data from another set where the target speed was low, but a practise strategy with an aggressive combine operation was used. The dotted lines show the average values of the respective bar plot. Frames 1a and 1b are from the same dataset. Frames 2a and 2b are from the same data set.
This is the first attempt of the eye-head to saccade to a moving target. Up to this time, it had only practised on stationary targets. Thus, it went straight to the first position of the target. Meanwhile, the target has moved to the left position. Each of frames 1-4 is from a run of 250 trials using a schema of 1 equilibrium point.

After much trial-and-error (usually 50-100 trials), this 1 equilibrium point, eye-head movement has progressed leftward, approaching the location where target next lands. What the system is learning is where to put the eye-head gaze corresponding to one starting location and one velocity vector of the target.

The schema now takes the eye-head just to the left of (i.e. ahead of) the 2nd position of the target. Once the eye-head settles, it repeatedly samples target movement. During this period, the target passes directly over the center of the gaze.

The 1 equilibrium pt. schema moves the eye-head so that the center of its gaze lands exactly where the target lands, and at the same time. Once settled, the eye-head continues to sample the target movement, though the schema cannot be improved.

**Figure 5.15:** In each of these frames the circles indicate positions of the target in time. The crosses indicate positions of the eye-head gaze in time. The target moves linearly and at constant velocity in the direction of the arrow. The textured lines indicate successive positions of the eye-head after saccade-like movements. In the frames circles do not appear at equal time intervals. Instead, circles appear at moments when the eye-head system needs to sample the location of the target. In each frame, the starting location of the eye-head gaze is the center cross, and the starting location of the target is at the lower right hand corner of the frame. When several images of the target occur closely spaced, as at the bottom of frame 3, the eye head system is sampling the location of the target at small intervals of time.
As in the previous figure, the eye-head first moves to the target's initial location. Since the schema has two CSE's, the eye-head then moves to the next location just occupied by the target. If the schema were longer, the eye-head would continue to be one jump behind the target.

Prior to learning a practise strategy, the 2nd eye-head location moves away from its initial position, and eventually merges with either the 1st or the 3rd location. This merge is due to minimization of a cost functional for the time-to-target.

Once the intermediate equilibrium pt. is gone, the remaining equilibrium pt. is well placed with respect to the anticipated target location. Only a small amount of additional search is needed.

Beginning with a 2-equilibrium pt. schema, the resulting, accurate, saccade-like movement requires much less search. Once a practise strategy is learned, the extra equilibrium pt. can be quickly removed, and a useful saccade results with very little search.

**Figure 5.16:** In each frame, the circles indicate successive target positions from right to left. The crosses indicate successive eye-head gaze positions from the center and following the textured lines. As before, the target moves at constant velocity. Thus, closely spaced target locations indicate more frequent visual sampling.
Starting with 3 equilibrium pts., this schema begins using equilibrium pt. values appropriate for a stationary target. Note that this would work just as well with previous learning at lower speeds.

With the first equilibrium pt. gone, less search is necessary for the remaining two to contact the target. This will obviously also work for longer schemas, effectively tracking the target.

As a final example, we with a 2 CSE schema. With similar initial conditions, different results can be obtained with different cost functionals. In this example, the position accuracy cost is applied to every CSE.

With the position accuracy cost applied to both CSE's, both are constrained to migrate toward target locations within their time window. A cost on total movement time is still needed to produce "compact" solutions.

Figure 5.17: As with the previous 2 figures, the circles indicate successive target positions from right to left. The crosses indicate successive eye-head gaze positions from the center and following the textured lines. As in previous examples, the target moves at constant velocity. The upper two frames correspond to one example where a schema of 3 equilibrium pts. combines away the first equilibrium pt., which produces an effective saccade sequence. The lower two frames correspond to a different example, the original two equilibrium pts. both migrate to later target locations in time.
The startup case just presented is shown in Figure 5.15, frame 1. As before, the expanding crosses are successive locations of the eye-head gaze, resulting from saccade-like movements. The expanding circles are successive locations of a single target moving at constant velocity. The successive positions of the target or eye-head do not necessarily represent equal time intervals. This should be clear from the target positions. The target always moves at constant velocity, so the different distances between successive target positions correspond to different intervals of time over which the eye-head samples the target positions (and velocities).

The diagrams in figures 5.15, 5.16 and 5.17 were generated by the simulation software itself, which outputs data according to events from the robot's perspective. In all the examples in the figures, the eye-head begins its gaze in the center of the box, and the target starts at the lower right corner and moves leftward. Usually, the nth target circle and the nth eye-head cross correspond to simultaneous events. This, of course, does not apply when there are fewer crosses than circles. In such cases, the eye-head has stopped moving and is continuing to sample target movements. Another exception occurs when there is a small cluster of several closely spaced circles. The cluster then corresponds to one eye-head position. The cluster represents a short interval loop where the eye-head is stationary, but rapidly sampling the target positions that constitute the cluster. This is the inner control loop of figure 5.18 (steps 5 through 7). Such a cluster of 4 successive target positions appears in frame 3. Note that there is also a cross within the cluster. The cross is where the eye-head came to rest after 1 saccade. The first, or rightmost, of the 4 circles is the simultaneous location of the target when the eye-head ended it's saccade.

Returning to the results, frame 1 of figure 5.15 shows the eye-head gaze occupying the starting location of the target, after the target has moved. After some stochastic search, in the form of practise trials, the eye-head saccade moves its landing spot closer to, but still lagging, the target's next sampled position. This state is depicted in frame 2. With additional stochastic search, the eye-head begins to land ahead of the target, as depicted in frame 3. In this new landing position, the eye-head detects that the target is approaching it. The eye-head then monitors the target's position to determine how close it gets to fovea center and how much time it took. With a little more stochastic search, the eye-head gaze lands directly on the target, as in frame 4.

Figure 5.14 summarizes some performance data for similar learning cases (using 2 or more CSE's to start). Frame 1a shows that on average the moving target task takes about 3 dozen trials to get to 4-8% position accuracy for a relatively slow target. Continued practise brings the system to 1-4% accuracy with a total of 74 practise trials on average. For a much faster target (2.8 times faster), it takes about 48 trials to reach 4-8% accuracy, and a total of 136 trials to reach 1-4% accuracy. The starting conditions were the same for all cases. Only stationary data was in the GRBF.

Figure 5.16 shows the effect of running a combine operator on short saccade sequences for this task. Frame 1 shows the first movement attempt, which gets its action vector from the stationary target data. What the diagram shows is that the eye-head movements put it one step behind the target. Frame 2 shows the eye-head schema some practise trials later. The eye-head position after the first saccade has moved in a way that shortens the total distance, and hence total flight-time, of the 2-saccade schema. In this case, the second saccade's equilibrium position did not move. This turns out to be a
frequent pattern, occurring in about 1/3 of the data for which we looked at intermediate phases of learning. Though probably not human-like, it does present the combine operation with a chance to eliminate the one equilibrium position. Frame 3 shows the result of the combine. The remaining equilibrium position now brings the eye-head to where its gaze lands just ahead of the target. A small amount of additional practise and frame 4 results, where the eye-head lands very close to the target (within 1%, though very slightly off).

Since the pattern in frame 2, of one equilibrium point staying fixed and the other equilibrium point merging with an adjacent one, is fairly common, the combine operation can be "discovered" by the system as valuable for this task. As such, it will get incorporated into a very simple practise strategy that starts the system off with a small number of CSE's per schema, and applies the combine operation either when progress slows down, or when two equilibrium positions begin to merge.

The effect of such a practise strategy on learning performance is shown in Figure 5.14, frame 3. Comparing frame 1b, for which no practise strategy was employed, with frame 3 shows substantially faster learning for the task. Such an improvement is contingent on an early application of the combine operation. This will happen only if the system has occasion to try the combine operation successfully early in the learning process. We will explore some developmental implications of this finding in the next section.

Figure 5.17 shows several other learning patterns from our data. Frames 1a and 1b show a similar pattern to that of figure 5.16. In this case, the first saccade was combined away from the sequence, leaving a primitive 2-saccade tracking schema. It turns out that if the cost functionals are averaged over all the saccades in a sequence, rather than simply applied to the last saccade in the sequence, then saccade-like tracking often emerges. For this emergence to be stabilized there obviously cannot be a simple cost penalty for long duration schemas. Time dependent costs have to be carefully balanced to minimize individual saccade durations, while permitting sequences of many saccades.

Frames 2a and 2b of figure 5.17 show another way in which this primitive form of tracking develops. In this run, the combine operation was either ineffective or never needed. Through stochastic search the system moved both equilibrium positions toward their respective target locations.

5.5.3 Explorations into reasons for sequences of short saccades in infants.

In the experiments of the previous sections we have touched on possible reasons why infants use sequences of short saccades to foveate objects when they are capable of making longer saccades (see chapter 2). We will summarize some possible reasons here, though we have yet to do the careful simulations needed to verify that our hypotheses are at least sufficient explanations for the phenomenon.

We have just seen something of a case for sequences of short saccades when the target object is in motion (section 5.5.2). Such sequences significantly decrease the amount of practise needed to foveate objects accurately, when the practise is enhanced with a practise strategy. In our data, we have seen reasonably good saccadic tracking immediately after the first CSE of the sequence was dropped out by the combine operator (in figure 5.14 compare frame 1b with frame 3).
The moving target data also suggest a very simple explanation for the infant pattern of learning to track low speed targets before higher speeds. The higher speeds require more search and therefore take longer to learn. In addition, tracking at higher speeds is more reliable if lower speeds are learned first, so that what is retrieved from the topographic map is closer to what is needed. This may explain the apparent preference infants have for slower moving targets earlier in development.

With regard to saccades to stationary targets, there is a difference in gestalt between our simulated sequences of short saccades and those observed in infants. Our sequences look like random walks (because they are), while the sequences that infants use are more directed. The reason for this is likely to be the result of how the topographic map is built up over time. Suppose at each location of the topographic map, an infant learns an assortment of saccades, whose magnitudes are restricted to a small range with small magnitudes, but whose directions include all 360 degrees. From every eye position, the infant would be able to make a short saccade in any direction. Thus, the pattern of learning could progress from short saccades at every location in the map to longer saccades from every location. The effect of such a pattern is that when a longer saccade is requested, than is present in the database, a shorter saccade of the same direction will be retrieved (this is what our 2^{d}-tree indexing will do). If a very long saccade is requested, then a directed sequence will be retrieved. Our simulated examples of learning to saccade a stationary target started from an empty database, not one already populated with data at every articulator location. When there's nothing in the database, a random value is returned, hence the random walk. However, we can simulate the progressive learning conditions we have just described. We will then see whether human-like saccade sequences are thus produced. This direction of research is planned for the near future.

The purpose of using a sequence of short saccades could be to increase the likelihood that each stochastically modulated sequence will get the eye near the target. Thus, the attempt will not only produce learning, but a useful eye movement.

Whenever the radius of search can be restricted, then the exploration needed for learning may not conflict as much with the need for an effective eye movement. Sequences of short saccades accomplish a smaller search field (i.e. a small radius around the landing zone of each saccade in the sequence) and eventually get the eye near the target. The tradeoff is speed. The sequence of short saccades being much slower than a single saccade. This learning strategy may serve another function as well. It may protect the eye from errant large saccades that could drive the eye into the limits of travel. In any case, the tradeoff of speed, in favor of eventually getting the eye to target is the basis of both the hypotheses for stationary targets and for moving targets.

5.5.4 Section summary

Consistent with the themes of earlier sections, the orientation we have taken toward eye-head movements is to look for learning methods that will produce useful movements early on in the learning process. By useful we mean moderately accurate, but sacrificing speed if need be.

In addition, the schema structures learned were unexpectedly variable and showed specific adaptations both within tasks (and hypothetical topographic maps) and between tasks. Systematic within task variation occurred in the stationary target task, where the
head-eye controller's variability required systematic compensation in the schema structure. Recall that for large head-eye movements a 2 saccade sequence (i.e. 2 CSE's) produced a shorter time to target than a single saccade. Another category of systematic within task variation could be seen in the stationary target data, where longer CSE sequences were appropriate for a schema early in the learning process, but much shorter (e.g. unitary) CSE sequences were appropriate for the same schema later in the learning process.

The moving target task also showed a variety of possible adaptations, perhaps because the cost functionals underconstrained the solutions. Nevertheless, a simple practise strategy, based on the combine operator, could significantly improve the rate at which the task was learned.

In addition, a change was made from applying the position error cost functional to the last CSE in the sequence to an average application of this functional to all CSE's in a sequence. This change induced longer CSE sequences in the moving target task, from which emerged a kind of tracking behavior.

5.6 Experiment 6: Learning practise strategies for visual and tactile tasks.

Much of chapter 4 is devoted to algorithms associated with the problem of learning practise strategies, while the basic visual and tactile grasp tasks are being learned. The software implementation of these algorithms is, for the most part, complete at this time of writing. However, we have not had the opportunity to run a large number of examples. As a result, we will summarize our experience so far, and leave the rest to future research.

In previous sections of this chapter, we alluded to a simple practise strategy that could be associated with the visual grasp task for moving targets. This practise strategy involves initializing empty schemas to some number of CSE's greater than 1, preferably 3 or more. It also involves occasional, but repeated, attempts to combine adjacent CSE's, thus putting later CSE's ahead of the target in an anticipatory position. The practise strategy becomes especially effective if it can learn to apply the combine operation to the first CSE of the schema, and if it can learn to do this as soon as possible after the creation of a schema, but not before the schema is filled with action vectors from the existing topographic map.

This is an especially easy practise strategy to learn because eliminating the first CSE of the schema in the moving target task almost always produces an immediate cost improvement. On the other hand, applying something like a split operation tends to make costs much worse immediately after the split. Consequently, the system very quickly learns to apply combine operations and not to apply split operations. On this basis alone, the system has dramatically improved its performance, because it frequently mixes combine operations in with mutation operations. When the combine works, the costs are almost immediately improved. When it does not work, such as if it is applied to the last CSE of a schema, performance suffers sufficiently soon afterward that the event history branching mechanisms detect this, and undo the combine. Thus progress is restored.\textsuperscript{13}

\textsuperscript{13} The problem of considering which variables are correlated with improvement and which are correlated with the opposite has not been adequately addressed in the thesis. We have finessed the problem by making only a small number of variables accessible for the context of practise strategy operations.
What tends to give the system difficulty, though, is to learn to initialize new schemas to larger numbers of CSE's. The difficulty is that, with a stationary GRBF network already in place, the learning performance in the moving target task is not bad even without a practise strategy, so distinguishing the long range effects of 1 CSE vs. 4 CSE's to begin with is not easy. This is one area where we have just not run enough cases yet to know whether the system can sort this out, or whether we need to consider additional mechanisms.

For the stationary target task, the very same comments apply, except that it is more beneficial if combine operations are not applied so early in the learning process. Longer schemas tend to help the system find the stationary target and foveate it within a very small number of trials. After that, combining away CSE's, except of course the last CSE, is usually beneficial. Recall that for large saccade angles, a split operation is needed to guarantee that the system can always find sequences of 2 or more CSE's. We have found that the simultaneous active presence of both the split and combine operations presents no problem. The system will try both and whichever works best at the moment will be kept. Eventually, it will find its way out of local minima, as with a more "standard" type of genetic algorithm.

If fact, these observations suggest that our definition of practise strategy should be altered in the following sense. Every genetic-like operator should be assigned some probability of firing. A practise strategy may sequence operators in the sense that it gives any particular operator a higher probability of firing under certain conditions or at certain times, as opposed to others (thus altering the weighted pattern of operator invocation over time). However, the learning process often seems to be well served when all the probabilities are non-zero.

For the reaching task to stationary targets\(^{14}\), the system does learn to use induction on the length of the equilibrium position sequence by beginning with 1 or several CSE's and splitting the temporal mesh whenever it reaches an impassable minimum in the cost functionals. As with the eye movement task, the difference between splitting and hence refining the temporal mesh vs. combining and hence coarsening the temporal mesh is a substantial difference with respect to the cost functionals. However, in the arm movement task, the effects of refining the temporal mesh are not immediate, but displaced in time. The system nevertheless learns the benefits of splitting CSE's because it has no other way of getting out of certain local minima in costs. On the other hand, combine operations are rarely beneficial in this task. Consequently, the system has not included combines in the practise strategy in any of the runs we have seen so far.

5.7 Temporal difference results

Here we will give brief mention to some negative results, and anticipate where we may find positive results in the future.

\(^{14}\) We have not as yet tried the reaching task to moving targets, though our simulation software should be able to do the task without modification. We eagerly await the results.

However, Drescher (1990) has addressed this problem in more detail in his thesis, and his results will be a starting point for our next efforts.
Early on in the formulation of this thesis, we tried to use temporal difference methods (see chapter 6) along with our usual complement of cost functionals in the base level learning algorithm. For example, in the arm task we would use TD along with a minimum-jerk cost functional. With this combination of functionals the the system would not show signs of convergence over thousands of repeated reaches to the same stationary target. The learning system appeared to oscillate between minimizing jerk, and accelerating the arm particularly quickly toward the target (i.e. minimizing time).

The reasons for this failure to converge become apparent when it is observed that the "discounted reward" used in TD (see chapter 6 or Sutton, 1990) produces a minimum-time cost functional. The computations employed in all forms of temporal difference methods are inherently minimum-time computations. Combining an inherently minimum-time functional directly with the minimum-jerk functional (which is more similar to minimum-energy) produces a cost surface with numerous competing local minima for time or for jerk. In other words, in a portion of the cost surface where time is the predominant functional being minimized, the jerk would usually be increasing and vice-versa. Our learning system was necessarily non-convergent, and we dropped the attempt. (Tesauro, 1991).

On the other hand, while TD does not work as a functional at the base level of our learning algorithm, we expect that it can work as a cost functional for practise strategy learning (the variation level). The aim of practise strategy learning is to learn new movements in the minimum number of trials. In other words, with respect to the rate of learning, a minimum-time method is appropriate and does not conflict with the minimum-jerk functional used at the base level. However, since we have not tried TD at this level, we can only speculate that it may work.

5.8 Summary

Taken together, our simulations have produced several developmental arguments in favor of the "equilibrium position hypothesis" (e.g. see Bizzi, Mussa-Ivaldi and Giszter, 1991; Bizzi and Mussa-Ivaldi, 1990; Bizzi, et al., 1984; Abend, et al., 1982). To summarize our earlier summary, the combination of equilibrium position control, high stiffness, and our basic learning algorithm are sufficient conditions for reaching movements to be learned while the arm is growing. The reaching movements so produced contact their target accurately and with low impact, even early in the learning process. Furthermore, the pattern of development of the reaching movements parallels the development of human reaching movements in certain characteristics: (1) discretization of the control strategy into a small number of movement units, (2) early non-linearities in the trajectory, (3) the appearance of the speed-curvature relationship within the non-linear trajectories, (4) the eventual development of low jerk, linear movements with a single movement unit for the reach. Furthermore, our model of infant reaching movements presupposes only those senses and perceptual abilities that are known to exist in infancy. That is, infants apparently do not watch the trajectory of their hands. Instead, they keep their eyes primarily on the target.

In our efforts to understand how the learning process interacts with growth, we found that if arm movements are to be kept on target, and if they are to have near zero final velocity, then only a small number of control points can be searched. This is
especially true if growth is very fast. Thus, the rate of growth may constrain the maximum number of movement units that will keep learning and growth in balance. In any case, the existence of movement units makes sense in the context of growth during the learning process.

In our growth experiments, we observed that as stiffness is increased the linearity of the movement will get worse before it gets better. Not surprisingly, we have found that stiffness can be helpful for learning to occur while the arm is growing. Thus, infant movements may be stiff and non-linear in order for learning to continue in the context of growth. This understanding ties together movement units, the degree of linearity, stiffness and equilibrium position control to account for the differences between infant reaching movements and adult movements.

Our experiments with the visual grasp tasks showed that these tasks are more complex than they might at first appear. In both the stationary and moving target versions of the tasks, a progression from a longer sequence of equilibrium positions in a schema to a shorter sequence (the opposite direction from the arm experiments) produces an effective schema at all phases of the learning process.

Such a progression parallels a curious pattern observed in infant eye movements where a sequence of short saccades (in one direction) are used where a single saccade would be possible. We found that this pattern makes sense when the learning task is considered in detail. For moving targets the pattern emerges if a stationary map of the task has been learned, and the map requires generalization for non-zero velocities. A short sequence of saccades based on the stationary map can be transformed into one accurate saccade to the moving target. This transformation can be sped up if a simple practise strategy using combine operations is used. Furthermore, if the cost functionals are modified slightly and applied to each equilibrium position in the sequence, then a saccade-based tracking behavior develops.

Particularly for the stationary task, long initial sequences of saccades help to get the eye to a target when the underlying topographic map is incompletely developed. These sequences, we hypothesize, trade speed for sufficient accuracy so that the eye movements are effective. Though we have not done the experiments, our simulation software was designed to do whole topographic map studies, not simply single trajectory studies. We will soon be able to see whether we can mimic infant eye movement development in greater detail, with the right initial conditions and constraints.

Finally, another way of framing the results of our moving target studies is that a form of target anticipation can develop, which is brought out and stabilized with a simple practise strategy. This anticipation occurs without any explicit form of planning. This suggests the possibility that some phenomena attributed to planning could be more simply explained through the application of variation operators that have served an anticipatory function in other contexts.
Chapter 6 Comparisons with related computational work

6.0 Introduction

In this chapter we will compare three categories of recent simulations that are related to this thesis. Studies in the first category treat learning of a similar task, in particular, a reaching task. Studies in the second category rely on an underlying theory of learning that is related to our own. In the first category are other simulations that learn reaching movements. We will consider several from a large variety of different approaches to learning reaching movements, including (Kuperstein, 1988), (Kawato, Maeda, Uno and Suzuki, 1990), (Edelman and Reeke 1982), (Massone and Bizzi, 1989), (Mel, 1989), (Berthier, Singh, Barto and Houk, 1992), (Jordan, 1990) and (Rimer, 1992). Clearly, this has been an active area of research within just the last few years. However, we will not be able to treat each approach in equal detail. Instead, we will explore several in detail and summarize the rest in a separate section.

In the second category are learning systems that are close, but not identical to ours theoretically. Included is an unsupervised learning model of Maes and Brooks (1990), which does not model reaching movements, but does contain an interesting approach to unsupervised learning. We will also consider a model of visual attention by Rimey and Brown (1990), because it is similar in spirit to our approach. Not included is a recent thesis by Gary Drescher (1991), which is an attempt to formalize some Piaget's work. We commented on this thesis in chapter 1. Time and space do not permit a careful analysis here.

Finally, we will conclude with a brief summary of work concerned with temporal aspects of the credit assignment problem, variously authored by Barto, Sutton, and Watkins (1990). While their approach to learning is quite different from ours (their work has grown out of the empiricist tradition in psychology, rather than the constructivist tradition that ours comes from), we have used a modified version of their class of credit assignment algorithms in several places within our system.

6.1 Other computational approaches to reaching and grasping
A Random Sampling Approach to Unsupervised Learning
Overview

Kuperstein (1988) has implemented an unsupervised, neural network model of one aspect of adaptive visuo-motor coordination of a simulated, multijoint arm with 5 degrees of freedom. Kuperstein's simulation has two eyes in a head+trunk unit, and an arm. The arm is modelled as several links positioned by antagonistic pairs of muscles. Setting the relative activation levels of such pairs of muscles is assumed to place the arm in a given position in space. Thus, Kuperstein does not attempt to model the dynamics of arm

---

1 We have restricted our attention to arm movements, rather than including eye and head movements, because the dynamic issues are essentially the same, and because there has been a lot of attention recently on learning arm movements. In any case, we have chosen to overlook some significant differences between eye-head movements and arm movements. We would expect that eye movements are dynamically simpler than arm movements, while eye movements in combination with head movements may not be any simpler, from the point of view of learning, than arm movements.
movements. Instead, his system acquires a composition of coordinate transformations from visual system coordinates, including the location of the target object on the retina as well as the direction of gaze w.r.t. the body, to articulator coordinates, which are activation variables that are monotonically related to joint angles.

**Learning algorithm**

Kuperstein's "vision" module does a limited form of edge detection, computing four orthogonal orientation directions by convolving an orientation kernel matrix with the 50 by 50 pixel intensity map. The contrast orientation map apparently produces probable edge orientation vs. position. Two edge orientation maps, one for each retina, constitute the input to an (unspecified) algorithm that determines the horizontal binocular disparity distribution.

The disparity map is the output of the simulated visual system, and is one of the inputs to a weight map. The other input is an eye-position or gaze map. What is noteworthy about this vision module is that it implicitly locates objects by virtue of a small cluster of disparities in an otherwise empty and noise-free visual environment. This highly simplified visual environment is probably essential for the system to reliably locate the object (a cylinder). Thus, this vision module does not truly constitute object location in the visual environment. Rather it is a black-box that provides simulated object locations for subsequent coordinate transformation maps.

The correlation between sensation and manipulation is developed in two phases, a learning phase and a generation phase. In the learning phase, self-produced, random motor signals are generated to explore a large range of arm postures (recall that these are static postures). Each time the arm is positioned, with the simulated object in hand, differences are determined between the actual motor signals keeping the arm in place, and the expected motor signals, which are generated via the function composition of the system's topographic maps. These maps are: (1) two sensory topographic maps, one used for locating the grasped object w.r.t the retina, and another for locating the eyes w.r.t the head+trunk unit; (2) a pair of connecting weight maps (one for the gaze direction and one for the location with respect to the retinæ), which are situated between the outputs of the vision module maps and the input of the motor map; and (3) a motor topographic map. A feedback controller keeps the arm in the position specified by the motor topographic map. These differences between the actual motor command, and the motor command produced by the composition of the sensory, weight and motor maps are used to drive changes in the weight maps. Thus, the difference between the randomly commanded position of the object, and the visually determined position of the object provide an error signal for backpropagation to modify the weight maps. In this way, the system is performing the calibration of a coordinate transformation from visual space to articulator space through multi-dimensional feedback.

Once the weight map has converged to stable values, the composition of the maps can be used to generate arm positions in response to placements of the test object within the visual field. This is, of course, the generation phase.
Conclusion
The idea of implementing coordinate transformations with multi-dimensional function approximation is a good one, and is somewhat similar to the work of Zipser and Andersen (1988). However, it is important to keep in mind that this is a very small part of the sensorimotor coordination problem. Though Kuperstein's system does learn the inverse kinematics of a 5 degree-of-freedom arm, it does not produce human-like trajectories as it does not learn the dynamics. Aspects of sensorimotor coordination that have not been addressed in this work include the problem of when visual attention and motor action are coordinated, how they are coordinated, and the dynamic aspects of motor control.

An Arm Trajectory Learning Study
Overview
Kawato, et al. (1990) attempted to construct a system that generates the dynamic control needed for minimum torque change arm movements. The Kawato, et al. (1990) system does trajectory formation, inverse kinematics and inverse dynamics for arm movements of a 2-link, 2-dimensional, simulated robot arm. Their system does all three parts of the movement problem in one integrated step, rather than 3 steps. Though their simulated system is a supervised learning system, it is mentioned here because it attempts to solve a problem that we are trying to solve, though in a different way.

Learning algorithm
Kawato's system is a multi-layered, feedforward neural net model with 5 by N units, where N is the number of time steps into which the control of the movement is divided. In order to simulate a movement, the N output units must be sequenced with delays of length n*delta_t for 0 <= n < N. The system has separate learning and trajectory generation phases.

During the learning phase, the network develops a forward model of the dynamics and kinematics. That is, there are N torque command inputs to the first layer, Tau_1, ..., Tau_N. Similarly, there are N hand position outputs from the 5th layer, X_1, ..., X_N. An intermediate layer, layer 3, encodes the sequence of N joint angle vectors, Theta_1, ..., Theta_N. The second and fourth layers are "hidden". Thus, the 5 layers of the network accept motor command input (layer 1), compute the forward dynamics (with intermediate layer 2), which results in a trajectory in articulator space (residing in layer 3), and finally another intermediate layer (4) participates in computing the forward kinematics, resulting in the trajectory in task space at output layer (5).

2 It is not clear whether biological organisms perform reaching movements in either the integrated way or as 3 modularly distinct steps. However, other aspects of Kawato's attempt probably preclude it from biological plausibility. In fact, an explicit optimization of the energy of the trajectory by relaxation is regarded by the authors themselves as not biologically plausible.
During the learning phase, back-propagation of trajectory errors was used to teach the system the forward model. Typically, 40 example trajectories with 100 sampling points each were used to train the system.

After the system learned the forward dynamics and kinematics models, the trained network could be used for arm trajectory generation. For trajectory generation, the system is given a starting point, and ending point and a via point. A relaxation procedure was used to generate a torque vector sequence that could drive the arm to a minimum torque-change movement. We will not describe this procedure, except to say that it is a Newton-like method applied to a function space, and that it is a regularization approach, using a discretized version of the minimum torque-change criterion as its smoothness constraint.

Conclusion

Kawato et al. claim that with a sufficiently complete learning of the forward model, the system should be able to generate any minimum torque change trajectory, including novel trajectories. The relaxation computation, however, is expensive and requires a fairly large number of iterations to reach a stable solution. This practical drawback may provide a suggestion that it is not plausible that biological systems do an explicit, on-the-fly optimization each time a trajectory is needed.

We have included the Kawato et al. work in our short review of related work because our system also learns to make minimum-jerk trajectories, but which has the following characteristics, which differ from the Kawato effort. First, our learning system does trial-and-error learning, rather than requiring a teacher, or some supplied desired trajectory. Second, it does not involve explicit optimization, or on-the-fly computation of a torque command sequence. Instead, the movement storage allows efficient "playback". Third, our simulation does not need separate learning and pattern generation phases. Instead, every experience of attempting a movement is an opportunity for learning. Fourth, our simulation is based on a fairly general purpose unsupervised learning mechanism.

An RBF Approach that Learns by Minimizing a Cost Function

Overview

A recent master's degree thesis by Jacob Rimer (Rimer, 1992) describes an assortment of learning experiments aimed at determining how appropriate gaussian RBF's are for learning minimum-jerk trajectories. In the last of these experiments he takes an approach similar to the approach adopted in this thesis. We will describe that experiment shortly.

To summarize the other experiments, Rimer used a 2-d, 2-link, simulated, kinematic robot arm. One experiment used RBF's to learn a function that takes as inputs the boundary points (beginning and ending positions and velocities in visual coordinates) of a movement and commands the kinematic arm through a minimum-jerk trajectory. The

---

3 Only two errors are available to the system during the trajectory generation phase. One of these errors is the difference between the final position of the hand and the intended final position, and the second error is the difference between the position of the intended via point and the closest trajectory position to the via point.
training set consisted of input-output examples of discretized minimum-jerk trajectories (parameterized by time) generated at many locations within the arm's workspace. In the test condition, when given a starting position (at 0 velocity) and an ending position in visual coordinates, the RBF network was quite good at producing straight, minimum-jerk trajectories from straight, minimum-jerk examples.

In another experiment, minimum-jerk trajectories made by humans were recorded, digitized and used as training data for an RBF network. These trajectories occurred in the coordinate system of the digitizing hardware. Example trajectories were made by two subjects, each producing several dozen trajectories scattered throughout the workspace. Unlike the previous experiment, the RBF network's output was a trajectory in the same coordinate system as the input (the plane of the digitizing tablet). The network was tested by providing as input the beginning and ending points of the movement. The network would then generate a (mimic) trajectory between those endpoints. Considering the likely variance to be found among the human example set and the relatively small number of examples, the quality of the mimicked result appeared to be reasonably good.

Both of these experiments were concerned only with the kinematics of movement. Thus, the network was essentially learning a coordinate transformation and a velocity profile, but not really learning the inverse dynamics of movements. Nevertheless, the results suggest that gaussian RBF's are appropriate for learning human-like trajectories.

Learning Algorithm

The experiment that comes closest to our approach, provoking a comparison, was an effort to learn, by trial-and-error, a parametric representation of a minimum-jerk movement between two points. At the start the system is given the initial and final positions, where the velocities \( v_i \) and \( v_f \) are zero, and a random trajectory that connects the two points. Using simulated annealing as the minimization technique, the system seeks to modify the trajectory in such a way that the accumulated jerk of the movement is lowered as much as possible. Again, it should be mentioned that Rimer's simulations use a kinematic arm, not a dynamic one as in our experiments.

The function computed by his network is \( F: T \to R^2 \) where \( T \) is a set of \( N+1 \) equal-interval time steps \( \{t_i | i = 0, ..., N\} \), \( t_0 = 0 \), \( t_f = t_N \). As outputs of the network, he chose \( v_x(t) \) and \( v_y(t) \). This contrasts with our choice to output equilibrium positions as a function of time and context. His startup example trajectory consisted of \( N+1 \) "examples", each of which was a triple, \( <t_i, v_x(t_i), v_y(t_i)> \).

Given that Rimer used gaussian RBF's, his network consisted of one input unit, whose input was \( t \), two output units corresponding to \( v_x(t) \) and \( v_y(t) \), and \( N+1+2 \) gaussian hidden units. The resulting computation is given by the following two equations.

\[
v_x(t) = \sum_{i=0}^{N} c_{ix} e^{-\frac{(t-t_i)^2}{\sigma}} + c_{startx} e^{-\frac{(-t\text{start})^2}{\sigma}} + c_{endx} e^{-\frac{(-t\text{end})^2}{\sigma}}
\]

\[
v_y(t) = \sum_{i=0}^{N} c_{iy} e^{-\frac{(t-t_i)^2}{\sigma}} + c_{starty} e^{-\frac{(-t\text{start})^2}{\sigma}} + c_{endy} e^{-\frac{(-t\text{end})^2}{\sigma}}
\]
\[ v_y(t) = \sum_{i=0}^{N} c_{i,y} e^{-\frac{(t - t_i)}{\sigma}} + c_{\text{start},y} e^{-\frac{(t - t_{\text{start}})}{\sigma}} + c_{\text{end},y} e^{-\frac{(t - t_{\text{end}})}{\sigma}} \]

What is similar to our approach is the method of search and minimization. At each iteration, a time interval and one velocity component are randomly chosen and randomly modified by a small amount. This small change in the trajectory will produce a change in the cost functional computed on the trajectory. If the change lowered the cost functional, then it was accepted as part of the best trajectory found so far. On the other hand if the change increased the cost functional, the change was rejected and the trajectory remained as it was before the change was tried. Unlike our algorithm, only one time interval of the trajectory was changed per iteration. Also unlike our algorithm, the temporal mesh was fixed, hence the need for giving the system a startup trajectory. In order to help the system out of local minima, trajectory changes that increased the cost functional were accepted with some probability.

We have included Rimer's cost functional, below. It combines a final position error constraint with the minimum-jerk constraint. Constraints to make the initial and final velocities and accelerations equal to zero were imposed directly on the initial and final units of the network.

\[ C = ||(x_f, y_f) - (\sum_{i=0}^{M} V_x(t_i) \Delta t_i, \sum_{i=0}^{M} V_y(t_i) \Delta t_i)|| + \lambda \sum_{i=0}^{M} (V_x(t_i) + V_y(t_i)) \]

As mentioned above, the search algorithm used for the minimization was simulated annealing. The \( N \) for his trajectories was 21. Rimer reports running this minimization process from 10,000 to 20,000 iterations. In all cases the trajectories were smoother than the initial trajectories and had roughly bell-shaped velocity profiles. However, he also reports many runs did not produce particularly smooth trajectories, and many runs resulted in the composition of two short, smooth trajectories (there was no constraint to prevent this). He did not comment on the linearity of the trajectories found. However, he did comment that the performance of the minimization process could be improved with a better choice of search than simulated annealing. We agree, and our results show that the combination of gradient descent, with genetic and stochastic search does much better. However, we would add that much can be gained by having the network generate equilibrium positions rather than velocities or torques.

**Conclusion**

Rimer has made a good case for using gaussian RBF's for human-like trajectory learning. We cannot help but agree with his approach to unsupervised learning. However, it is our task to point out the differences between his work and ours.

The differences that come most readily to mind are (1) our system is able to "discover" a first trajectory, (2) via variation operators our system can adjust the temporal and spatial mesh according to task demands, (3) our simulated robot arm is dynamic and the learning process is not degraded by arm dynamics, (4) we found that network output of choice was that of equilibrium positions, (5) there was no counterpart practise strategy
learning in Rimer's system, (6) our database techniques support learning multiple trajectories and generalizing among them, this is not a feature of his unsupervised learning algorithm, (7) we have incorporated various heuristics to limit the possibility of oversampling, and (8) our system is not specifically tailored to one problem.

6.2 Related theoretical approaches
A Goal-based Approach to Unsupervised Learning
Overview
An example of an approach to unsupervised learning, that is based on the same notion of goals as suggested in this proposal, is a study by Maes and Brooks (1990). The authors implemented a 6-legged, insect-like robot, that learns to walk (statically stable walking) by learning to coordinate the actions of its legs. Using an algorithm described below, the robot learns to walk by trial-and-error attempts. Initially, the robot has difficulty even standing up, let alone moving forward. Gradually, its movements become more coherent, and in the process it discovers the tripod gait used by most insects.

Learning algorithm
Each leg has its own, independently running controller. The system has 2 goals (in our terminology) that provide success or failure signals for the leg controllers. The goals are (1) to move forward, and (2) to keep the ventral surface of the robot off the ground. A goal-positive signal is available to all the leg controllers when both of these conditions are true. A goal-negative signal is available to all controllers when either of these conditions is false. Each leg process tries to determine under what conditions it maximizes goal-positive signals, and minimizes negative or goal-negative signals, while measuring how relevant (a statistical measure, discussed below) its actions are to the two active goals.

Additional information available to the leg controllers is as follows. (1) There is a vector of binary perceptual conditions. That is, each sensor is either on or off, and one vector encodes all sensors. In this case, the sensors indicate the current positions of all of the legs. (2) There is a set of behaviors, from which each controller can select one at a time. A behavior is a set of processes involving sensing and acting. Each behavior has a precondition list, which is a conjunction of predicates on the sensation vector. In Maes and Brooks' paradigm, a behavior becomes active when its precondition list is satisfied as a logical expression. When active, the executing processes can produce somewhat complex interactions with the environment. Each leg controller maintains and alters its own precondition list. For the walker experiments, 12 behaviors were used, one swing-leg-forward behavior for each leg, and one swing-leg-backward behavior. Apparently, all the legs were driven slowly toward the rear when in stance phase, producing a net forward movement when leg swings are properly coordinated.

The learning task for each leg controller is to incrementally change the precondition list of the behaviors for its leg, so that in time, behaviors only become active when they are both relevant and reliable in relation to the positions of the other legs. A behavior, together with its precondition list, is relevant when it is positively correlated to

---

4 The measure of relevance is based on the Pearson product-moment correlation coefficient.
the goal-positive signal, and not positively correlated with the goal-negative signal. A behavior, together with its precondition list, is reliable if it receives consistent goal-related signals. That is, if the probability of receiving a goal-positive signal when the behavior is active is close to 1 or 0. Similarly, for goal-negative signals when active (i.e. it could be reliable in an effective or an ineffective sense). Each leg controller monitors and updates the relevance and reliability of its behaviors with respect to their current precondition lists, while the robot organism is trying to walk. There is no separate training phase.

Relevance is used to determine the probability that the behavior will become active with the current precondition list. That is, the more relevant a behavior is, the more likely it will become active when the precondition list is satisfied. The less relevant a behavior is, the more likely it will be that experimental changes in the precondition list are made. Low relevance initiates systematic search through precondition space for an altered list. A behavior that is not relevant has little chance of becoming active.

The control strategy is as follows. Behaviors are grouped by actuator. At each timestep the selectable behaviors in every group are determined (i.e. those behaviors which are not yet active and whose preconditions are fulfilled). For each of these groups, one or zero behaviors are selected probabilistically according to (1) the relevance of the behavior, (2) the reliability of the behavior, and (3) with a bias toward activating behaviors for which there is very little experience in the current condition. The selected behaviors are then activated.

It should be mentioned that goal-related feedback is immediate in this model. There is no explicit notion of actions in time. In fact, that is why it is possible to have goal-negative signals. In most situations, it is easy to determine whether a goal has been achieved, but very difficult to determine when actions are leading a system away from its goal. The problem of estimating or evaluating actions, whose consequences will be felt later in time will be discussed in a later section.

Conclusion

There are several aspects of this work that are noteworthy. First, the notion of goal in Maes' algorithm is one of an internal process that monitors the sensory inputs and provides a signal indicating whether actions taken are promoting or defeating the goal. Second, Maes' algorithm is oriented toward developing successful patterns of action, whereas the reinforcement learning algorithms (discussed below) are somewhat oriented toward building state-models of the world. Third, there is no delay between an action and its goal-related signal, suggesting a direction for future work on this algorithm. Fourth, there is another somewhat subtle difference from other work that is related to its action rather than state orientation. Drescher's (1989) and Watkins' (1989) (discussed in a later section) work is based on a conviction that knowledge is an internalized copy of states.

---

5 A curious aspect of Drescher's thesis is that his expressed intention was to construct a computational model of Piaget's theory. However, he has done this from an epistemological position quite incompatible with Piaget's. What he has actually done is to model the developmental events that are described in Piaget's work within an AI style paradigm. Piaget's theory of the processes underlying these developmental events was overlooked.
relations in the outside world. The implicit ontology of knowledge in the Maes and Brooks paradigm is that knowledge is the capacity to interact with the environment in the service of the organism's goals. These are very different assumptions about representation, and in fact characterize the difference between many cognitive psychology paradigms (together with many AI paradigms as well), which are founded on the former definition of knowledge, and the neo-Piagetian (and constructivist) developmental psychology paradigms, which are founded on the latter notion of knowledge (e.g. see Gruber and Voneche, 1977).

**Visual Attention as a Sequential Process**

**Overview**

Rimey and Brown (1990) have begun to investigate the problem of the explicit representation of task-dependent attentional sequencing of eye-movements. For this work, they have employed hidden Markov models (HMM), which are, loosely speaking, teachable and probabilistic finite state automata. Actually, they have augmented the HMMs (AHMMs) so that they can be modified by visual feedback during execution.

**Learning algorithm**

In their implementation, eye movements are commanded by the outputs of hidden Markov states. Since these outputs are determined probabilistically, adding a new output with a corresponding likelihood of occurrence is relatively easy. The probabilities of the other possible outputs for that state are adjusted accordingly. This gives the AHMM used by Rimey and Brown a form of flexibility in executing stored eye-movement sequences. Salient features, located either in the periphery or in the fovea, are able to override the stored sequence when these features are particularly strong, and when they are located in slightly different positions from those that the stored sequence would predict.

Eye movement sequences are stored into AHMMs from examples using an algorithm due to Rabiner (1989). As examples, Rimey and Brown used recorded human eye movements for a single task, or artificially constructed examples entered using a mouse controlled pointer on example images.

The task independent measures of salience that were tried consisted of a weighted sum of several image intensity based computations and several different edge algorithms. The authors did not regard their choice as anything more than a temporary means of implementing salience. They suggested that salience for attention is probably task dependent and needs to be studied.

**Conclusion**

In this paradigm, visual feedback modifies the output of the hidden Markov state, rather than modifying the state or adding a state. Thus, Rimey and Brown's formalism can alter an existing eye movement strategy somewhat, but cannot derive a new one. This leaves open the question that we would most like to study, which is how eye movement strategies originate during the process of solving a visual problem. Modelling attention-related processes with probabilistic automata seems to be a reasonable starting point into this area of research.
6.3 Related approaches to temporal credit assignment

Temporal Difference Algorithms

Overview

Learning how to interact with an environment in an unsupervised way entails performing actions whose consequences may not be available until some time after the action is taken. This is especially true for the sensorimotor problems that are the subject of this proposal. There is a growing body of research on this topic, which has acquired the unfortunate name, reinforcement learning. See for example (Barto, Sutton, and Watkins, 1990) (Barto, Sutton, and Anderson, 1983) (Sutton, 1988, 1990), (Watkins, 1989). These authors have combined the use of the theory of sequential decision processes (e.g. Markov decision processes) with dynamic programming methods.

Learning algorithm

Typically, sequential decision systems involve a decision maker that interacts with a discrete-time stochastic dynamical system. At each time step, the decision maker monitors the system's current state and selects an action. After the action is taken (i.e. at the next time step), the decision maker receives a certain amount of reward (which in some systems may be zero until the goal is actually achieved), which depends on the action and the current state. The system then makes a transition to a new state, which is determined by the current state, the action, and random disturbances. Upon observing the new state, the decision maker chooses another action and continues in this manner for a sequence of time steps. The objective of the task is to determine the relation (which in some cases is a function) for the decision maker to use in selecting actions. This relation between states to actions is called a policy. The decision problem, then, is to determine a policy that maximizes the expected return.

Let \( p \) be a policy, \( x_t \) be the system state at time \( t \), \( a_t \) be the action taken at time \( t \), and \( r_t \) be the reward received for taking action \( a_t \) from state \( x_t \). By these definitions, \( p(x_t) = a_t \). Suppose that after the action is taken the system is in state \( x_{t+1} \). Suppose also that \( 0 \leq g < 1 \), then the discounted return from time \( t \) is defined to be

\[
r_t + g r_{t+1} + g^2 r_{t+2} + ... + g^n r_{t+n} + ...,
\]

and the expected discounted return when following policy, \( p \), is denoted

\[
E_p[ \sum_{t=0}^{\infty} g^t r_{t+1} \mid x_0 = x ].
\]

In order to compute optimal policies, dynamic programming methods compute sequences of different types of evaluation functions. An evaluation function for a given policy assigns to each state of the dynamical system the expected value of the return, assuming that the system starts in that state (assuming the dynamical system has the Markov property), and follows the policy when choosing actions. For policy, \( p \), discount factor \( g \), and initial state \( x \), the evaluation function \( V_p(x) \) is defined as
\[ V_p(x) = E_p[ \sum_{t=0}^{\infty} g^t r_{t+1} | x_0 = x ]. \]

For each state, the evaluation function predicts the return that will accrue when the system is started in state \( x \) and run indefinitely into the future. Evaluation functions provide the means of biasing the value associated with a state with rewards that may occur at a distant time. That is, evaluation functions provide a means of assigning values to decisions based on the distant consequences of those decisions.

Strictly speaking, dynamic programming methods require that the decision problem be completely specified. The temporal difference algorithms proposed by Sutton (1988) and Watkins (1989), on the other hand, construct approximate evaluation functions for a given policy, but in the absence of knowledge of the transition probabilities of the dynamical system, and in the absence of the function determining the expected rewards. In addition, several algorithms proposed by Watkins (1989) determine approximate evaluation functions while the policy is being improved.

To get a limited sense of how this is done, let \( \Phi(x_t) \) be a feature vector representing the state \( x_t \), and let the estimated evaluation function be

\[ V_t(x_t) = f(w_t, \Phi(x_t)), \]

where \( w_t \) is the weight vector at time \( t \) and \( f \) depends on the approximation technique used. The temporal difference algorithm updates the weight vector using the rule,

\[ w_{t+1} = w_t + k [ r_{t+1} + g V_t(x_{t+1}) - V_t(x_t) ] \frac{df}{dw_t}(\Phi(x_t)), \]

where \( k \) is a step size parameter. For a justification of this procedure, consult (Barto et al., 1990) or (Sutton, 1988).

It should be apparent that popular approximation techniques, such as neural nets using back-propagation of errors are easily applied to evaluation functions. Since an evaluation is a function of state, policy and reward, any smoothness constraint helps to propagate the effects of temporally distant rewards to evaluations earlier in time.

As an example of an application of these methods to a simple learning task, we will summarize a simulation implemented by Watkins (1989) as part of his thesis. The task of the simulation is a simple route finding problem. The state space of the problem is a square in the Euclidean plane, centered at the origin. Within this state space are two objects: a small square target, and a robot whose task it is to learn how to navigate to the target from any position in the state space.

At any point in the state-space, the set of possible actions the robot can take are moves from the set of two dimensional vectors \( [\Delta x, \Delta y]^T \). The reward function is a step function. There is a large reward for entering the target area. The reward for any other movement in the state space is zero.
The simulation begins with a policy that is the zero vector everywhere in the state space. Similarly, the evaluation function is set to zero everywhere. Actions are chosen by calculating the policy action for the current position, and then adding a randomly generated deviation vector. That is, $a_t = p(x_t) + dev_t$. Deviations are necessary, because, in order to improve the policy, actions must be taken that are different from those of the policy. To give the robot well distributed experience in the state space, it is trained by numerous runs where it is started from random positions within the space. In this version of reinforcement learning, there is exactly one action associated with each state.

Both the policy function and the evaluation function are adaptively estimated during the course of the simulation by a temporal difference algorithm. Initially, the robot's actions are a random Markov walk. Consequently, after a few runs, the early entries in the policy function constitute a random vector field. The evaluation function after a few successful runs is smooth and flat throughout most of the state space, except near the target, where it looks like a small hill (the range of values is [0,1]), centered over the target area. After a few thousand successful runs, the policy becomes a vector field with all vectors pointing straight at the target. The evaluation function becomes a smooth approximation to a cone, with maximum values over the target, and minimum values at the most distant locations in the state space from the target.

**Conclusion**

This example was included in the discussion because it is similar to the problem of unsupervised learning of reaching movements in several important respects. First, it involves learning a trajectory when the success-related feedback only occurs when the target is more-or-less accidently encountered. This is similar to the developmental problem that an infant is faced with. Second, the result of the learning is analogous to a topographic map whose input is a sensory topography and whose output is a vector field. Considerable additional mechanisms need to be hypothesized, however, in order to apply Watkins' techniques to the more dynamic problem of discovering straight, minimum-jerk movements.

Finally, we should mention that we do not subscribe to the notion of rewards or punishments supplied to a passive organism by the environment. Instead, we have hypothesized that internal goal processes provide part of the mechanism of assessment of actions. However, the mathematical formalisms discussed above do not care where the reward signals come from. Consequently, these temporal difference algorithms are equally applicable within our meta-theoretical framework, as they are to Barto et al.

**6.4 Summary**

We have presented a small sampling from a growing collection of studies of unsupervised learning of arm movements. Kuperstein's (1988) project learned coordinate transformations (the inverse kinematics) by uniform random sampling of the transformations. Rimer (1991) used an approach to unsupervised learning of the inverse kinematics of arm movements using simulated annealing to perform a stochastic search toward a minimum in the cost functional for jerk. The idea of not using a "sample movment", but having the system generate movement attempts stochastically, being guided by the accumulated jerk functional is similar to our basic adaptive loop, as applied
to the reaching task. There are, however, many differences between our approach and his, among them are that our system is able to "discover" a first trajectory, while Rimer's system required an initial "solution" to the reaching problem; via variation operators our system can adjust the temporal and spatial mesh according to task demands, while Rimer's system uses a fixed temporal mesh; our simulated robot arm is dynamic, Rimer's is kinemated; and we used equilibrium positions as the output of the network, while Rimer used the velocity vector as the output of the network;

Maes and Brooks (1990) implemented a form of active, goal-directed learning of leg coordination for a 6-legged walker. The learning occurred as a result of trial-and-error experiments while trying to walk. Their goal-directed approach is quite similar to ours, though they have focussed on different problems. Their focus has been on coordinating (i.e. selecting and sequencing) multiple goals where the actions taken by the system are "fixed action patterns." Our focus has been to put learning into "action patterns" in order to make them adaptive rather than fixed. In other words, we have focussed on problems with one goal, but in more detail. Similarly, their sensory "conditions" have discrete binary values, while our sensory dimensions take on a closed interval of values for each dimension.

Barto, Sutton, and Watkins (1990) have been studying the temporal aspects of unsupervised learning, using a combination of Markov decision processes and dynamic programming. Their orientation is that the environment provides reward or punishment signals that guide learning. However, they have not yet hypothesized mechanisms that can select and interpret events in the environment into positive or negative signals. Nevertheless, their formalisms for temporal credit assignment are useful, regardless of their metatheoretical orientation. Recall from chapter 5 that we tried to use TD for the reaching task, but it conflicted with the minimum-jerk functional. The required alternative approaches to credit assignment based on hypothetical affect mechanisms.

Finally, Rimey and Brown (1990) studied selective attention as a sequential behavior, using hidden Markov models. In their study, an acknowledged ad hoc definition of visual salience drove the learning. This is a different approach from the others in the discussion in that an assumption about the nature of the representation is used to guide the learning. Becker and Hinton (1989) have suggested, somewhat similarly, that unsupervised learning algorithms can be developed by defining functions that characterize the internal representations, without requiring knowledge of the desired outputs of the system. To contrast these approaches with ours, in our developmental approach, learning to apply specific senses at specific times within a task is an outcome of the developmental mechanisms of learning employed to learn the task. In other words, attention emerges from the solutions, the schemas, that the system discovers are appropriate to specific goals. To say this yet another way, attention is a consequence of developmental mechanisms. Therefore, understanding attention is a developmental problem. We plan to pursue this notion in future research.
Conclusion

With this thesis we have attempted to put forward a metatheoretical framework for constructing theories about development (chapter 1); to select two milestones in infant development for study (chapter 2), namely visual and tactile grasp; to construct a model of the developmental processes sufficient to account for the infant's accomplishment of these two milestones (chapters 3 and 4); to implement the model of the developmental processes with computer algorithms; and to demonstrate that the model works through simulation experiments using the computer implementation with simulated or real robot hardware (chapter 5).

The metatheoretical framework we formulated is constructivist, neo-Piagetian up to a point, and includes a significant role for affect in learning. Indeed, the definition of affect and its relation to development are novel aspects of our metatheory and theory. Simply put, affect mechanisms are those mechanisms that tell us how we are progressing in relation to our goals. Our claim is that human developmental processes cannot be adequately understood without including affect in the understanding. Another way to put this is that models of (unsupervised) learning require the mechanisms of affect, if the models are models of human learning. Giving affect a central and necessary role in cognition is a major departure from any of the dominant trends in psychology, including Piaget's.

Another departure we have made from the dominant trends in psychology is to abandon the notion of representation as a copy of an external reality, and the notion of representation as a state-machine simulation of an external reality. Either of these definitions of representation has its roots in empiricist epistemology. Instead, we have adopted a constructivist epistemology similar to Piaget's genetic epistemology. In this epistemology, knowledge is the ability to interact with an external environment in a way that achieves one's goals. It is important to state that both goal mechanisms and affect mechanisms are regarded as internal to the organism under study, not aspects of the external environment. In other words, we are departing from empiricist inspired epistemologies by putting goal and affect mechanisms inside the head, and regarding knowledge as something which is constructed by these mechanisms in the course of interacting with the world, rather than regarding knowledge as something which is outside the head to be copied inside. Furthermore, the something that knowledge is is an interactively competent control structure, a schema in Piaget's use of the term. Thus, in our epistemology, the problem of representation does not directly involve structures of the world. Instead, representation is instrumental, and the problem of representation is the problem of how control structures get built that support the organism's goals. Simply put, the problem is how the organism's developmental machinery writes programs that use its senses and motors as inputs and outputs for interacting with the world. Such programs terminate when the current goal is satisfied (much of the time).

These metatheoretical departures from conventional psychology put us on new ground, and within a very different field of possible psychological theories. Having made these departures, the thesis went on to explore a theory of development somewhat motivated by Piaget's theory, but which departed from Piaget's theory whenever there was a clear need to do so. The resulting basic adaptive loop architecture captured Piaget's
notion of adaptation with its component processes of assimilation and accommodation, as well as the goal-directed, active nature of infant interactions. However, the basic adaptive loop also incorporated some first guesses at several mechanisms of affect. These mechanisms are needed to determine which parts of a schema (i.e. which parts of a control structure) are moving toward the goal (and therefore should be kept) and which are not (and should be discarded).

Still more mechanism was needed to make Piaget's notion of adaptation work. We needed to find a mechanism for modifying schemas (control structures) and keeping track of which modifications work and under which circumstances. These new mechanisms constitute an entire additional level to the system architecture, which we called the variation level of the architecture. At the variation level, there are variation operators, which are genetic-like transformations of control structures. When variation operators are sequenced, they become a control structure in their own right, which we called a practise strategy. When a practise strategy can be generalized from successful instances then the adaptive architecture has a primitive ability to learn how to learn.

With a Lisp implementation of this multi-level architecture, we were able to explore some questions posed in the experimental literature on infant development. We demonstrated that it is possible to learn to make linear and smooth reaching movements by trial-and-error without example movements to imitate (reaching develops very early, so imitation is probably not involved), and without tracking the hand by eye to guide the hand during the movement. The target location is established visually, but proprioceptive control and tactile termination are sufficient for reaching to develop. In other words, infants apparently do not watch the trajectory of their hands as they learn to reach for things. Instead, they keep their eyes primarily on the target, and this is sufficient for arm movements. Furthermore, we showed that there is a developmental advantage to generating reaching movements by sequences of equilibrium positions.

The reaching movements so produced contact their target accurately and with low impact, even early in the learning process. Furthermore, the pattern of development of the reaching movements parallels the development of human reaching movements in certain characteristics: (1) discretization of the control strategy into a small number of movement units, (2) early non-linearities in the trajectory, (3) the appearance of the speed-curvature relationship within the non-linear trajectories, (4) the eventual development of low jerk, linear movements with a single movement unit for the reach.

Our developmentally inspired architecture is able to learn to reach while the arm is growing. During the experiments that were used to verify learning during growth we discovered that under certain conditions, a speed-curvature relationship (as noted in both infant and adult arm movement literatures) emerges. In our growth experiments, we observed that as stiffness is increased the linearity of the movement will get worse before it gets better. Not surprisingly, we found that stiffness can be helpful for learning to occur while the arm is growing. Thus, infant movements may be stiff and non-linear in order for learning to continue in the context of growth. However, the conditions of high stiffness, equilibrium position control and fast growth together produced local speed minima in the reaching trajectories that corresponded exactly with local curvature maxima in the trajectories. This understanding ties together movement units, the degree of linearity,
stiffness and equilibrium position control to account for the differences between infant reaching movements and adult movements.

Our experiments with the visual grasp tasks showed that these tasks are more complex than they might at first appear. In both the stationary and moving target versions of the tasks, a progression from a longer sequence of equilibrium positions in a schema to a shorter sequence (the opposite direction from the arm experiments) produces an effective schema at all phases of the learning process.

Such a progression parallels a curious pattern observed in infant eye movements where a sequence of short saccades (in one direction) are used where a single saccade would be possible. We found that this pattern makes sense when the learning task is considered in detail. For moving targets the pattern emerges if a topographic map for stationary targets has been learned, and the map requires generalization for non-zero velocities. A short sequence of saccades based on the stationary map can be transformed into one accurate saccade to the moving target. This transformation can be sped up if a simple practice strategy using combine operators is used. Furthermore, a slight change in the cost functionals for the task can produce a saccade-based tracking behavior.

Particularly for the stationary task, long initial sequences of saccades help to get the eye to a target when the underlying topographic map is incompletely developed. These sequences, we hypothesize, trade speed for sufficient accuracy so that the eye movements are effective, but much slower than adult eye movements. We plan to study the development of infant eye movements further.

Finally, another way of framing the results of our moving target studies is that a form of target anticipation can develop, which is brought out and stabilized with a simple practice strategy. This anticipation occurs without any explicit form of planning. This suggests the possibility that some phenomena attributed to planning could be more simply explained through the application of variation operators that have served an anticipatory function in other contexts.
References


