NEURAL MODELS OF DEPTH PERCEPTION IN FROGS AND TODS

A Dissertation Presented
By
DONALD HENRY HOUSE

COINS Technical Report 84-16

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

Doctor of Philosophy

September 1984

Department of Computer and Information Science

ABSTRACT

NEURAL MODELS OF DEPTH PERCEPTION

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Cybernetics as the Science of Sensory-Motor Coordination

Cybernetics is a term which seems to have as many meanings as there are persons prepared to use it. For example, the following definition appeared in the 1960 edition of Webster's New World Dictionary, College Edition:

> cy·ber·net·ics (sī'bẽr-net'iks), *n.pl.*[construed as sing.], [<Gr. *kybernetes*, helmsman; + *-ics*], a science dealing with the comparative study of complex electronic calculating machines and the human nervous system in an attempt to explain the nature of the brain.

This definition differs widely from the meaning originally given it by Norbert Wiener [1948], who intended it to encompass

> ...the entire field of control and communication, whether in the machine or in the animal...

A recent tendency is to use the term interchangably with *system dynamics*, implying that the study of any system whose components are interrelated in a complex way is a cybernetic study. We embrace the notion that cybernetics has firm roots in the brain, computing, and systems sciences but none of these definitions is entirely consistent with ours.

In our view cybernetics is the science which addresses the problems of sensory-motor coordination, whether in biological, robotic, or even hybrid systems.[1]

1 Wiener's example of control of an anti-aircraft gun was just such a hybrid system, involving not only radar, ballistics and aerodynamics but also the actions of the the gunner and the pilot.

This view stops short of the very loose current usage that allows one to apply the term cybernetics to ecological, economic, and social systems but avoids the overly narrow definition given in Webster's. In practice our definition is closest to Wiener's.

Because within our view of cybernetics there are no boundaries between artificial and natural sensory-motor systems, cross-disciplinary studies are invited. The study of natural systems, systems which are so successful at performing sensory-motor tasks that have proven difficult to implement in machines, holds promise of indicating elegant solutions to practical problems in robotics. Conversely, when a solution to a sensory-motor problem has proven successful in an artificial system it becomes a rich source of analogy in the study of natural systems.

Depth Perception as an Issue in Cybernetics

Our definition of cybernetics implies a point-of-view about the relationship between sensory and motor processes. Sensory processes and motor processes cannot and should not be put into separate categories [Arbib, 1972]. Sensation is for a purpose. It is used to adjust action. Action in turn affects perception. Action and perception are two inseparable parts of a closed cycle [Neisser, 1976; Arbib, 1981].

In this dissertation we take an approach to the study of depth perception that reflects the cybernetic point of view. If depth discrimination is imbedded within a cycle of action and perception then it cannot be viewed as simply an information-organizing process separate from action. If, for instance, a toad exhibits size-constancy[2] in its preference for prey then do we assume that the toad estimates the distance of the prey and then determines its size or do we assume that a side effect of the toad's prey acquistion strategy is that prey of a certain size are preferred? The cybernetician must certainly consider the latter.

Our point of view from cybernetics does not allow us to treat depth perception as a passive process, concerned only with assigning a depth value to each point in visual space. Instead, it must be viewed as part of a dynamic process by which an animal (or machine or human) achieves a particular goal. If the goal is navigation through the environment then what is required is quite different from what is required if the goal is to catch a passing fly. Depth perception, to the extent that it is an identifiable function of the visual system, may take as many different forms as there are distinctly different functions which require its use.

Purpose of the Dissertation

In this dissertation we present a sequence of models of the extraction of depth information from visual data. Our study differs from other recent studies in several ways. First, rather than using the mammalian depth perception system to guide our exploration we use data from the literature on frogs and toads. Second, we are concerned with how depth information from multiple cue sources can be integrated. Finally, our point of view that perception is action oriented required us to pay

[2] An animal exhibiting size-constancy appears to make judgements about visually presented objects that reflect the object's real size rather than subsumed visual angle.

careful attention to behavioral, as well as anatomical and physiological, data.

The purpose for making these models is twofold. Since the models are constrained to conform to existing knowledge of the visuo-motor system of frogs and toads, they have the potential for making contributions to the understanding of visuo-motor coordination in these animals. By building and experimenting with computer simulations of the models we have taken a first step in the development and testing of algorithms for artificial systems. Because our goals span two disciplines, our perspective embraces but is not limited by the view espoused by Riss [1968] that

> [a]dequate theory must have relevance to behavioral, physiological and morphological scrutiny and must be susceptible to improvement through each of these empirical avenues.

We have paid careful attention to data from experimentalists and hope that, in turn, our models will be the subject of experimental testing. We also hope that our models will prove useful in the design of control systems for robots even if frogs and toads prove to be insistent on refuting our speculations.

Since we wished to forge links between two divergent disciplines we took care to simulate our neural models on the computer. Experiments done with the computer simulations allowed us to go beyond mere plausibility arguments by providing concrete demonstrations of the efficacy of the models. They allowed us to analyze the sensitivity of the models to variations in parameter settings and environmental configurations. This analysis, in turn, led to many concrete predictions about animal behavior which are now open for verification by experimentalists. Further, the computer simulations represent operating algorithms that can be implemented in robotic systems. Experiments with these simulations provide important data concerning the tuning and expected performance characteristics of these algorithms.

Frogs and toads were chosen for this study for several reasons. First, they exhibit ballistic behavior. Their prey-catching sequence consists of periods of apparent inactivity followed by a series of movements that seem to be preplanned. Changes in the environment after initiation of overt activity do not seem to affect how that sequence is carried out. Once a toad begins to stalk and strike at a prey it will not stop even if the prey target is artifically removed; the sequence, including swallowing and wiping the mouth after the strike, is uninterrupted [Ewert, 1980]. Ballistic behavior is remarkably akin to the actions of preprogrammed robots. A second reason for choosing frogs and toads is that they are vertebrates but with simpler brains than those of many other vertebrates. Therefore, it may be possible to draw parallels to visual functioning in higher vertebrates without having to confront their greater complexity [Scalia & Fite, 1974]. Finally, frogs and toads exhibit highly developed depth vision but also appear to utilize fairly simple rules when using depth information to coordinate their behavior [Collett, 1982]. Along with depth, these simple rules employ only readily available visual cues. Thus, they are of the sort which could conceivably be implemented in a robotic system.

Organization of the Dissertation

Each chapter of the dissertation includes enough background material so that the chapter can be read as an independent unit. The price of this policy is redundancy. Redundant material has, in the main, been confined to the introductory section of each chapter. It may be skipped, without penalty, by whoever has the interest and stamina to read the dissertation from start to finish.

In Chapter 2 we review current depth perception models and those aspects of the visual system of frogs and toads which appear to play a role in depth perception. This chapter has three goals. The first of these is to familiarize the reader with the theoretical and experimental data relevant to depth perception in frogs and toads. The second goal is to point out the shortcomings of current depth perception models with respect to the problem in frogs and toads. The final goal is to prepare the reader for some of the conclusions reached during the development and testing of the models in the next three chapters.

In Chapter 3 we describe a model that demonstrates a way in which earlier cooperative depth models might be extended to better address the data from frogs and toads. We call this model the "cue interaction model" since it uses monocular depth cues from lens accommodation to assist in the selection of binocular depth cues from local disparity matching. The use of accommodation to assist in the disambiguation of the binocular matching problem reduces dependency on densely populated images, continuity of surfaces in space, and depth cues from binocular convergence.

In Chapter 4 we present another, quite different, model of depth perception. This model, the "prey localization model," also addresses the disambiguation of binocular matching by the use lens accommodation. However, unlike the "cue interaction model," this model does not use monocular depth cues *per se*. Instead, quality of image focus is one of the controlling variables in a feedback loop that adjusts lens focus in order to simultaneously select and localize a single prey target. Since this model is limited to determining the spatial position of a single prey it is not as ambitious as the "cue interaction model," which attempts to provide a complete depth-map of the animal's binocular visual-space. However, in an important sense it is an advance: it gives a detailed explanation of how lens focusing can be coupled with a binocular system to select a particular prey from among several, and accurately estimate its depth.

Chapter 5 provides a synthesis of the models presented in the previous two chapters. Here we argue that depth perception in frogs and toads may not be a single process. We observe that the process underlying the "cue interaction model" is most appropriate for locating barrier objects, whereas the process underlying the "prey localization model" is (as its name implies) most appropriate for locating prey. Starting with these observations, we construct a preliminary model of a lens accommodation controller that could support both processes.

Finally, in Chapter 6 we conclude with a synopsis of the most important findings of the modeling work, further animal studies suggested by the work, and the implications of the work for robotics.

The appendices provide the detailed description necessary to allow replication of the results presented in the dissertation. Appendix A shows how the mathematical definition of the models was developed. It also describes the structure of the simulation algorithms, and explains how parameters were chosen for the simulation experiments. Appendix B provides a description of how the optics were simulated. Appendix C describes the software system used to run the computer simulations. It contains both a user's guide, and a programmer's guide to assist those who might wish to use this system for their own simulations.

CHAPTER II

DEPTH MODELS AND THE VISUAL SYSTEM OF FROGS AND TOADS

Summary

In this chapter we review the reasons why previous theoretical models do not provide an adequate explanation for the depth perception process in frogs and toads. We begin with a survey of earlier depth models and a review of experimental evidence from behavioral, anatomical, and physiological experiments relating to depth perception in frogs and toads. We conclude by contrasting the assumptions made in the models with observations about the animals. We note that all of the previous depth models based on stereopsis are restricted to the consideration of binocular matching on a pair of static images, and that their purpose is to produce a depth-mapping from the image pair. These models also depend upon assumptions concerning vergence and image granularity. We show that these restrictions and assumptions are not applicable to the depth resolution problem in frogs and toads.

Previous Models of Depth Perception

Strategies for obtaining depth information from optical data fall into three main categories: 1) optic flow algorithms, 2) autofocus algorithms, and 3) stereoscopic algorithms. Optic flow algorithms utilize a time-sequence of images from a single sensor (camera or eye) together with sensor movement data to reconstruct depth from parallax between consecutively scanned images. Some of the more recent work on optic flow algorithms is described in Prager and Arbib [1983], and Lawton [1984]. Autofocus algorithms attempt to adjust the lens of the sensor to maximize some measure of image focus. In this sense they are not depth algorithms at all, but once the lens is adjusted depth can be inferred from its setting. A short review of the various autofocus algorithms is contained in Ligthart and Groen [1982]. Selker [1982] describes a hardware implementation of such an autofocusing system. Stereoscopic algorithms address the problem of obtaining depth from binocular images by computing disparities between matching regions on the two images.

Neither the optic flow nor the autofocus algorithms were deemed appropriate to the study of depth perception in frogs and toads. The optic flow algorithms all entail the use of a moving sensor, but frogs and toads are able to determine depth while remaining stationary and exhibit no tracking eye movements [Autrum, 1959; Ewert, 1980]. Existing autofocus algorithms all use a global measure of image focus to adjust the lens. Therefore, they are most useful with sensors having a narrow field of view. However, the monocular visual field of frogs and toads exceeds 180° [Fite and Scalia, 1976].

The stereoscopic algorithms are also not directly applicable to the depth perception problem in frogs and toads, but the reasons for their unsuitability are more subtle. The remainder of this chapter provides an analysis of these subtleties.

Studies of human depth discrimination in random-dot stereograms have provided most of the incentive for research in stereoscopic depth vision. Julesz [1971] showed that humans could successfully fuse a stereo pair of random-dot patterns so that if a central rectangle in one image (selected arbitrarily, and not defined by edges in the image) were shifted with respect to that in the other image the observer would see this central pattern as either hovering above or sunken into the surrounding pattern. This ability to binocularly match regions of a pair of images without the use of high-level monocular features shows that pattern recognition is not necessary for binocular depth resolution.

Nelson [1975] developed a neurophysiologically grounded theoretical model that explained Julesz' results. Dev [1975] and later Marr and T. Poggio [1976] developed computational models following the principles outlined by Nelson. Their models are based on cooperation and competition in spatially distributed arrays of neuron-like elements. In these models a single disparity (or depth) is assigned to each retinal position by choosing the most likely estimate at that position. In both of these models similar assumptions are made about the input image pair. The most important assumption is the predominance of continuity – the observation that objects in the world tend to have continuous surfaces which vary only gradually in depth except at object boundaries. As a result, most neighboring image points share similar disparities. Thus, a reasonable computational scheme is to weigh a depth estimate more heavily when it is similar to estimates at neighboring image sites. The algorithms are made more efficient if it is assumed that the eyes are verged upon a spatial point near the image region that is being analyzed. This assumption assures that the range of disparities is small and that, in the absence of other cues, a smaller disparity is more likely than a larger one.

Hirai and Fukushima [1978] extended these earlier cooperative/competitive schemes by introducing explicit cues from vergence to assist in disambiguation of binocular matching. In addition to using neighboring estimates in their weighting scheme, they also use a global matching process which helps to assure that 1) a point in one image is matched by only one point in the other image, and 2) that the ordering of the matches is consistent.

Trehub [1978] also extended the earlier cooperative/competitive algorithms. He proposed that performance could be improved by using results of a local pattern coorelation, in addition to biasing from neighboring estimates, to assure proper binocular matching. His scheme requires the additional assumption that a fine-grained pattern of stimulation be available at each local matching site.

Amari and Arbib [1977] present a theoretical study of the general class of cooperative/competitive "selection" algorithms. Their work includes several theorems which relate parameter settings to model equilibria.

More recently, modelers have exploited spatial-frequency tuned channels to guide the binocular matching process in a hierarchical way. Models by both Marr and T. Poggio [1979], and Frisby and Mayhew [1980] match features of spatial-frequency band-passed versions of the images, so that the matches obtained from the low-frequency versions of the image can be used to bias the matching process in the higher-frequency components. The models differ in that Frisby and Mayhew follow previous models in using a cooperative scheme whereas Marr and T. Poggio do not.

The common purpose of all of these depth models is to produce a depth mapping of the image under consideration. To do this, the models employ only binocular depth cues and resort only to the information contained in one static stereo-pair of images. An assumption (stated or otherwise) that is made in all of these models is that binocular convergence has succeeded in producing images in which the range of binocular disparities is both small and centered about zero. It is also implicitly assumed that the images are clearly focused. Finally, all of these models depend upon a fine-grained pattern of visual stimulation being available to the matching algorithm. In order to evaluate the applicability of these assumptions to the depth perception system of frogs and toads we will first examine selected background material on the visual system of these animals.

Depth Perception in Frogs and Toads

In this section we present data from behavioral, anatomical, and physiological studies that relate to the perception of depth by frogs and toads. Behavioral results demonstrate how depth information is utilized, and indicate that depth information comes from both binocular and monocular cue sources. Anatomical and physiological findings indicate those brain structures that seem most likely to be involved in the perception of depth. Our review traces the flow of visual information from the retinas to the brain-stem motor centers. We also present the results of several studies that correlate the anatomical and physiological units with behavior.

The role of depth perception in detour behavior

When they are involved in catching prey or avoiding predators, frogs and toads use depth information in surprisingly sophisticated ways. For example, when approaching a prey located behind a barrier, frogs and toads exhibit size constancy for both the prey and the barrier objects [Lock & Collett, 1979; Ingle, 1976]. They also demonstrate the ability to measure the distance between barrier and prey [Lock & Collett, 1980], and they can remember the position of a prey and orient towards that position even after the prey is no longer visible [Collett, 1982].

The choices that toads make in detouring around a barrier to get to prey give clues as to how they utilize depth information. Lock and Collett [1979, 1980] demonstrated that when confronted with a paling-fence barrier between it and its prey a toad chooses either to detour around one of the fence ends, to push through the fence in an attempt to reach the prey directly, or to ignore the prey. Later, Collett [1982] showed that in choosing among these three options toads make decisions that depend upon barrier position and length, distance and position of prey relative to the barrier, and the placement and width of gaps in barriers. In all cases, these judgements of length, width and distance appear to be based on estimates of the actual dimension and not on a simple measurement of subsumed visual angle.

Toads are able to simultaneously locate two barrier surfaces in depth. Fig. 1 shows the results of several of Collett's experiments where he presented toads with various barrier/prey configurations. Fig. 1a shows a toad faced with a distant unbroken fence between it and its prey. In this case 75% of the approaches were aimed at one of the fence ends, with only 25% aimed directly at the prey. Fig. 1b shows a nearer fence with a central gap. Here, approaches were aimed almost entirely at the gap. However, when two fences, similar to those above, were presented simultaneously, as in Fig. 1c, the toads showed a response that was unlike either response to the individual fences. Instead, it was more nearly an average of the two responses, with about 50% of the approaches being directed towards a fence end and the other 50% directed at the gap in the front fence. One explanation for these results is that the toad actually sees the two fences in separate depth planes. Collett showed that toads can measure the distance between two fences by running the same experiment but this time varying the distance between the two fences. He found that the percentage of approaches aimed at the frontal gap increased as the distance between the fences increased.

The ability to discriminate the distance between fences is not based upon measuring the difference in angular distance between the fenceposts. To prove this Collett tried several configurations involving a single fence with a central segment whose palings were more closely placed than those in the rest of the fence. Results were similar to those obtained when all fence posts were evenly spaced. Approaches still either detoured around the fence or or were directed towards the prey. The toads never oriented towards the position where the spacing between the fence posts

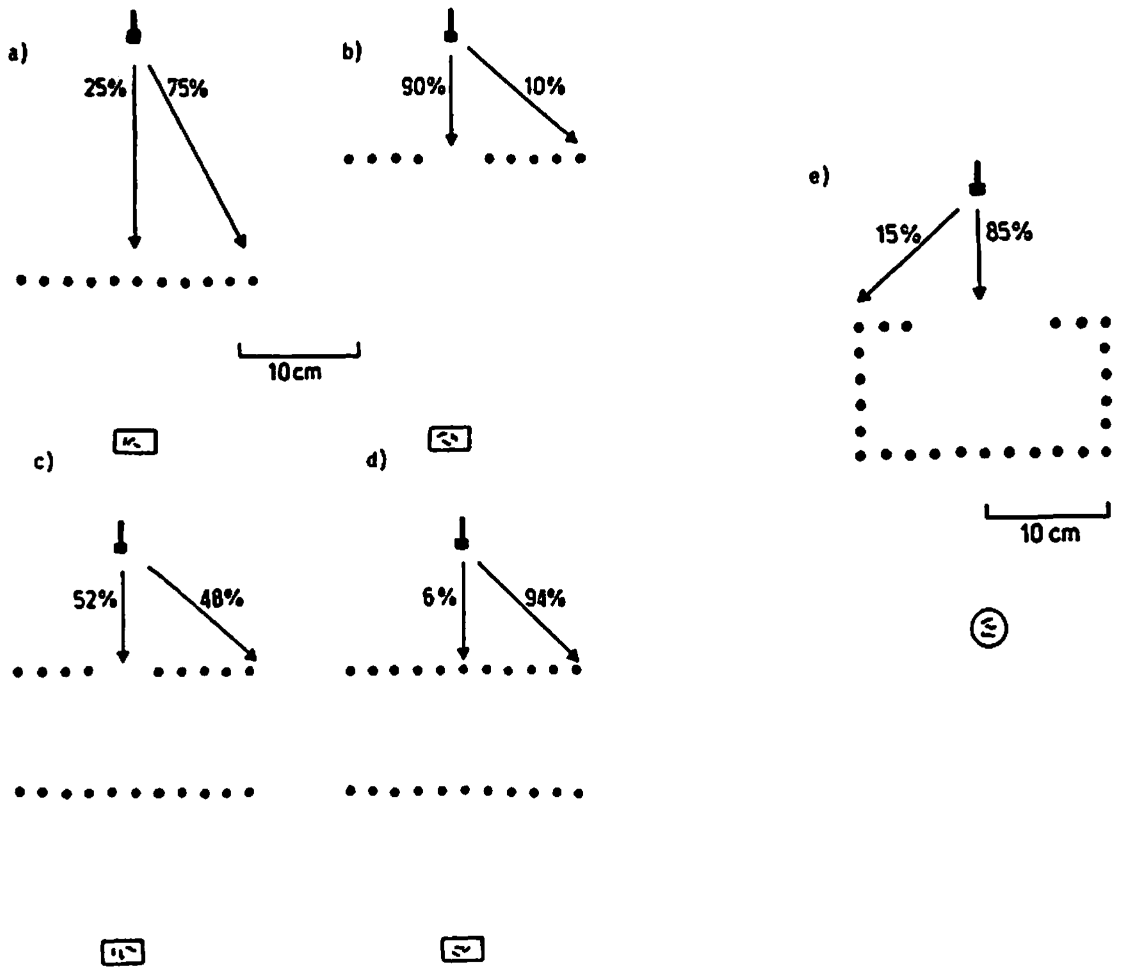


Fig. 1 - Prey/Barrier Experiments.

Histograms of the initial orientation response of a toad presented with various prey/barrier configurations. Configurations are shown in top-view. Rows of dots represent paling fence barriers, large circles indicate prey position, and the inverted T's represent the position and orientation of a toad. Distances are to the scale shown on the diagram. Percentages indicate the number of trials on which the animals elected to head directly towards the prey versus the number on which they elected to detour around either of the fence ends. Reprinted by permission from Collett [1982].

changed.

Toads are able to measure the distance between a barrier and a prey. Lock and Collett [1980] showed that when a toad is confronted with a prey behind a fence-like barrier, the distance between the toad and the fence does not affect the percentage of approaches that detour around the fence. As long as the distance between the prey and the fence is held constant it does not matter how far the configuration is from the toad. However, the distance between the prey and the fence has a major effect upon approach direction preference. Moving the prey closer to the fence increases the number of direct approaches, whereas moving the prey further from the fence increases the number of detours. Collett [1982] showed that this effect is maintained even when there are two fences and the prey is behind the rear fence.

During prey-catching, toads plan a route to their prey which they can follow whether or not the prey stays in view during the approach. Fig. 2 depicts another experiment by Collett [1982] that demonstrates this fact. In this experiment a paling fence was interposed between the toad and its prey. Behind the fence was a T-shaped opaque barrier that allowed the toad to view the prey only when the toad's long axis was aligned with the channel forming the leg of the T. Once the toad began its detour approach, the prey was hidden by the barrier. Nevertheless, turns around the barrier ends were directed towards the vicinity of the prey. If the toad elected to continue its approach after rounding the barrier it tended to head towards the position of the prey.

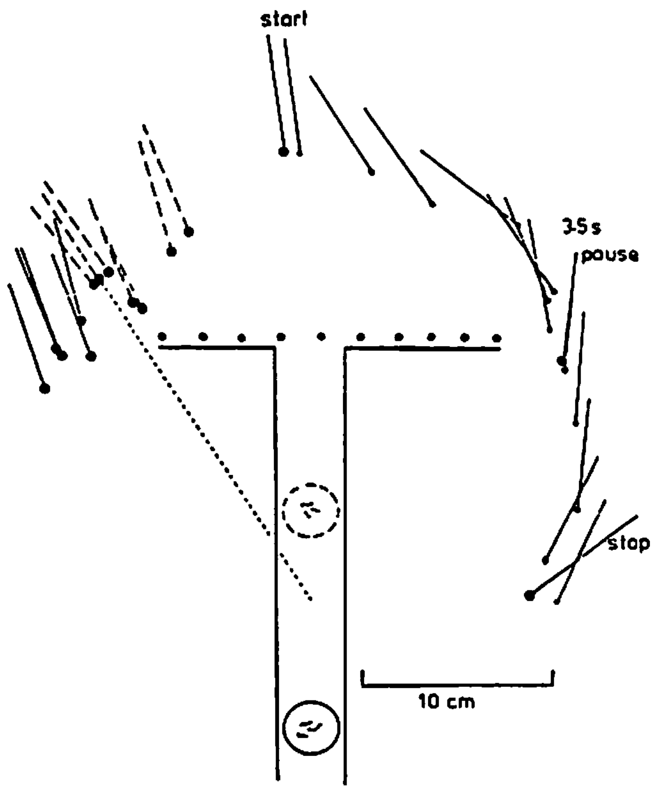


Fig. 2 - Toad Trajectories.

Results of several experiments involving a toad's approach to prey behind an occluding barrier. Solid lines to the right indicate the orientation of the body axis of a toad and its snout position (dots) at intervals along its path towards the prey. For this case, prey objects are those shown enclosed within the solid circle. After it begins its movement, the T shaped opaque barrier prevents the toad from seeing the prey. Solid lines to the left show the orientation of the toad's body axis, for several trials, during its pause at the fence end. Dashed lines are similar but for prey positioned within the dashed circle. These data make clear the toad's ability to 1) extract depth information from its visual world, 2) maintain a short-term memory of this depth information, and 3) integrate this memory with some notion of its own body movement. Reprinted by permission from Collett [1982].

Monocular and binocular depth cues

Frogs and toads use both monocular and binocular cues to determine depth. When both types of cue are available, the binocular cues are preferred. Ingle [1976] demonstrated that monocular toads are able to snap accurately at prey located throughout the visual field of their intact eye, although there is some tendency to undershoot targets placed contralateral to the midline. Fig. 3 summarizes this result. The presence of significant error only in the periphery, where lens resolving power is least, led Ingle to hypothesize that lens accommodation provides the monocular depth cues. Later, Collett [1977] found that toads are also able to estimate depth monocularly. He confirmed Ingle's hypothesis about accommodation by showing that a concave lens placed in front of the eye of a monocular toad will cause it to undershoot prey. However, Collett also showed that binocular toads utilize binocular depth cues almost exclusively for estimating the depth of targets in the binocular field. He found that lenses have only a weak effect on depth estimation in binocular toads but that base-out prisms placed in front of their eyes produce significant undershooting. He was able to closely match his experimental data with a formula that gives a 94% weighting to binocular cues and only 6% to monocular cues. Collett's experiments and findings are summarized in Fig. 4.

A more recent series of experiments, carried out in Roth's laboratory, offers strong confirmation of the results of Collett and Ingle. Jordan et al. [1980] tested the depth resolving ability of toads (*Bufo bufo*) after application of drugs to alter the tension in the lens accommodation muscles. They found that in binocular toads both Atropine (a muscle relaxor) and Miotic (which produces strong muscle

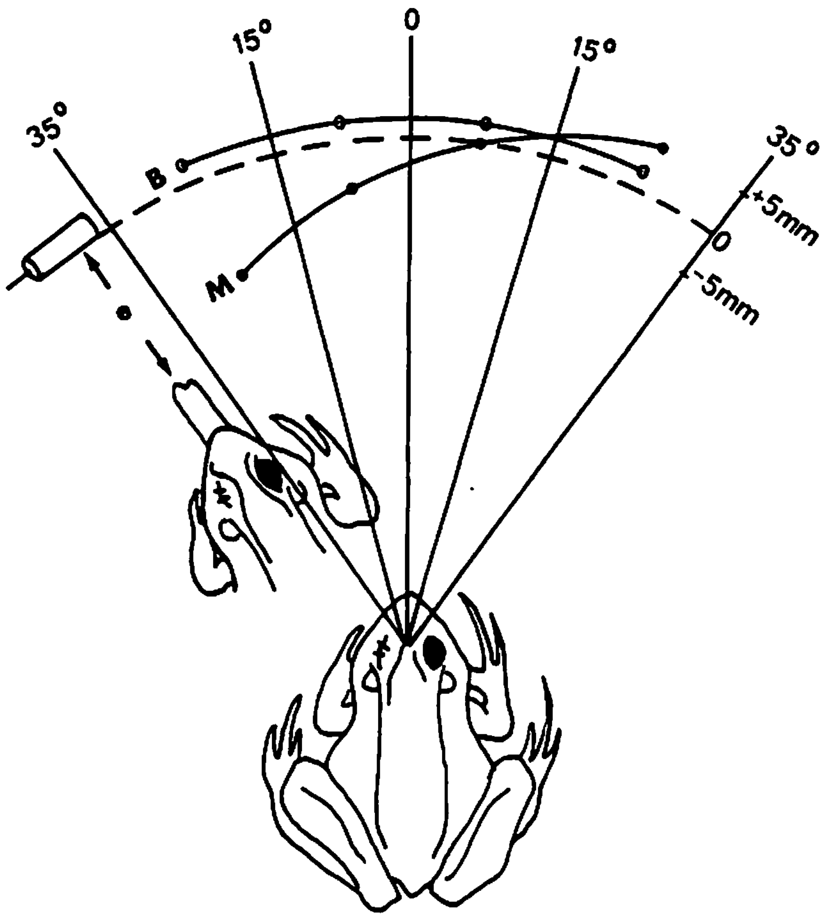


Fig. 3 - Snapping Errors After Monocular Blinding.

Measurements are from cine-recordings. Solid lines mark the deviation of the tip of the wet tongue mark from the near edge of the cylindrical target (dashed line). Line B shows the mean tongue extension for binocular frogs. It slightly overlaps the stimulus across the entire rostral binocular field. Monocular frogs (Line M) fell short of the target for locations 15-35 degrees from the midline, within the contralateral visual field. Errors within the ipsilateral visual field are not significantly greater for monocular animals than for binocular animals. Thus, it may be concluded that monocular information is sufficient for depth perception in frogs. The data allude to the use of accommodation for monocular depth information since the error is worst in the periphery, where the accommodative power of the lens is the least. Reprinted by permission from Ingle [1976].

Fig. 4 - Effect of Prisms and Lenses on Depth Perception.

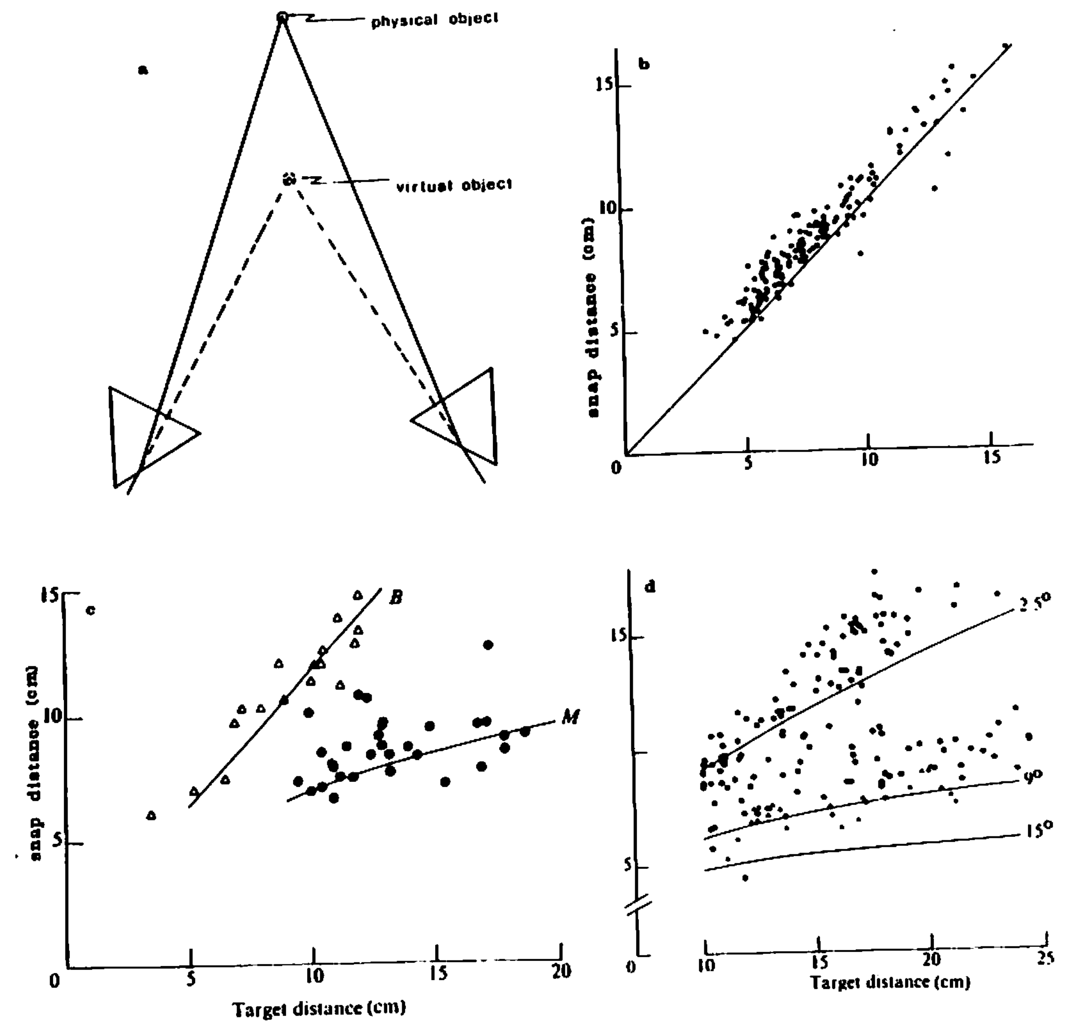
(a) The effect of base-out prisms on binocular depth perception. The angular shifts in retinal position of the physical object result in its being perceived as nearer than it actually is (virtual object).

(b) Scatter-plots from tests showing that both binocular and monocular toads are able to accurately judge prey distance. The oblique line represents the exact distance of the target from the animal. Filled circles show the results of trials with binocular toads and the open circles with monocular toads. Snapping distance for nearly all trials shows the overshoot seen by Ingle (see Fig. 3) with frogs.

(c) The effect of concave lenses on snapping distance. Open triangles are for trials with binocular animals, filled circles with monocular animals. Line B shows the snapping distance expected (with a small correction for the prismatic effect of the lenses) if the change in focal length due to the lenses has no effect upon depth estimation. Line M shows the snapping distance predicted by the change in focal length. Binocular toads show very little change in snapping distance, whereas monocular toads are strongly influenced by the presence of lenses. These trials indicate that binocular cues dominate monocular ones in estimating depth for snapping. They also indicate that monocular depth cues are derived from lens accommodation.

(d) The effect of prisms on snapping distance. 2.5 degree, 9 degree, and 15 degree lines show the snapping distance predicted by consideration of the effect of these three prismatic strengths upon visual parallax. Filled circles represent trials with 2.5 degree prisms, open circles with 9 degree prisms, and triangles with 15 degree prisms. Distortion of depth estimation is progressively greater with increasing prismatic strength, confirming the dominance of binocular depth cues over monocular ones.

Reprinted by permission from Collett [1977].


a
physical object
virtual object
b
snap distance (cm)
c
B
M
snap distance (cm)
Target distance (cm)
d
25°
9°
15°
Target distance (cm)

contractions) have only a small effect on perception of the depth of prey. However, monocular toads will severely undershoot prey after treatment with Atropine and overshoot after treatment with Miotic.

Anatomy and Physiology

This section describes the neural structures and pathways that seem most likely to form the substrates for depth vision. Our intent is to provide a selected survey of the results that bear on the problem of depth perception. Following this description we will present the results of experiments attempting to link the neural data with behavioral data.

The eyes

In both frogs and toads the eyes are outward facing but structured to give the animal vision over nearly the entire superior hemisphere. The inferior hemisphere is also well covered, particularly in the frontal area. Fig. 5 (from Fite & Scalia [1976]) shows this field for both *Rana pipiens* and *Rana catesbieana*. A significant area of binocular overlap is apparent. In *Rana pipiens*, for instance, the frontal binocular field covers more than 60 degrees.[1]

Frogs and toads do not make vergence eye movements and likewise have no distinct foveal area of the retina. Eye movements are not made to fixate or track objects in the visual field [Autrum, 1959; Ewert, 1980]. The small eye movements which do occur, such as those made during opto-kinetic nystagmus [Montgomery et

[1] See also Grobstein et al. [1980].

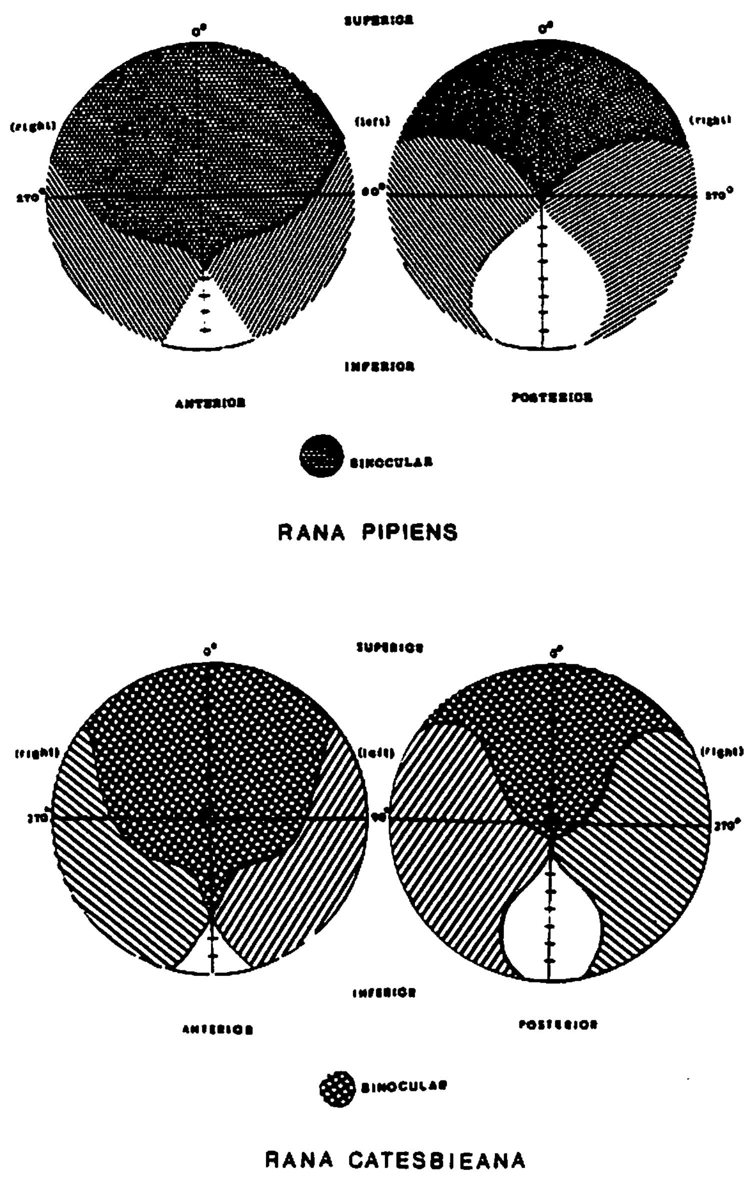


Fig. 5 - Visual Field Maps in Frog.

Perimetric maps of anterior and posterior fields of view as measured for *Rana pipiens* and *Rana catesbieana* showing the right and left eye's visual field and the region of binocular overlap. Reprinted by permission from Fite and Scalia [1976].

al., 1982], are probably related to image stabilization. Without the ability to binocularly converge the eyes there is no means by which a selected portion of the image can be fixated at a known point on the two retinas. Consistent with their lack of vergence, the eyes of frogs and toads have no distinct foveal area. Instead, photoreceptors and the ganglion cells contributing to the optic nerve are distributed relatively evenly (compared with higher vertebrates) over the retinal surface, as shown in Fig. 6 (from Fite & Scalia [1976]).

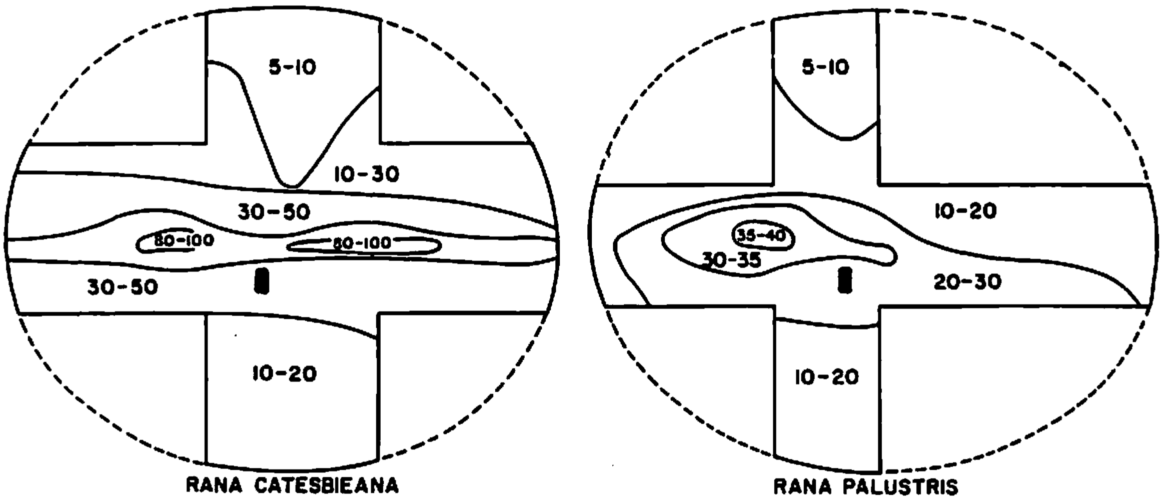


Fig. 6 - Retinal Receptor Density Maps.

These maps of receptor density show that there is no specialized foveal area on the frog's retina. Instead, recptor density is relatively evenly distributed over the entire visual field. This is consistent with the fact that neither frogs nor toads have the capability for making the vergence eye movements necessary for visual fixation. Reprinted by permission from Fite and Scalia [1976].

Optic nerve projection sites

The optic nerves cross at the chiasm, sending most fibers to contralateral diencephalic and mesencephalic brain regions [Scalia & Fite, 1974]. Dense retinotopically organized projections extend to the optic tectum with smaller projections to several thalamic nuclei and to the basal optic nucleus. These projections are shown in Fig. 7. The tectum is the major retinal projection site. Projections to tectum are monocular, coming almost entirely from the contralateral retina. However, within the thalamus the corpus geniculatum, nucleus of Bellonci, posterior thalamic nucleus, and uncinate pretectal nucleus all receive binocular input. Ipsilateral projections from the binocular visual field to these thalamic nuclei are as dense as the contralateral projections [Fite & Scalia, 1976]. Projections to sites in the dorsal anterior thalamus and the basal optic nucleus are monocular. Beneath the most anterior portion of the tectum, and thus not shown in Fig. 7, is the large-celled pretectal nucleus which receives a monocular projection.

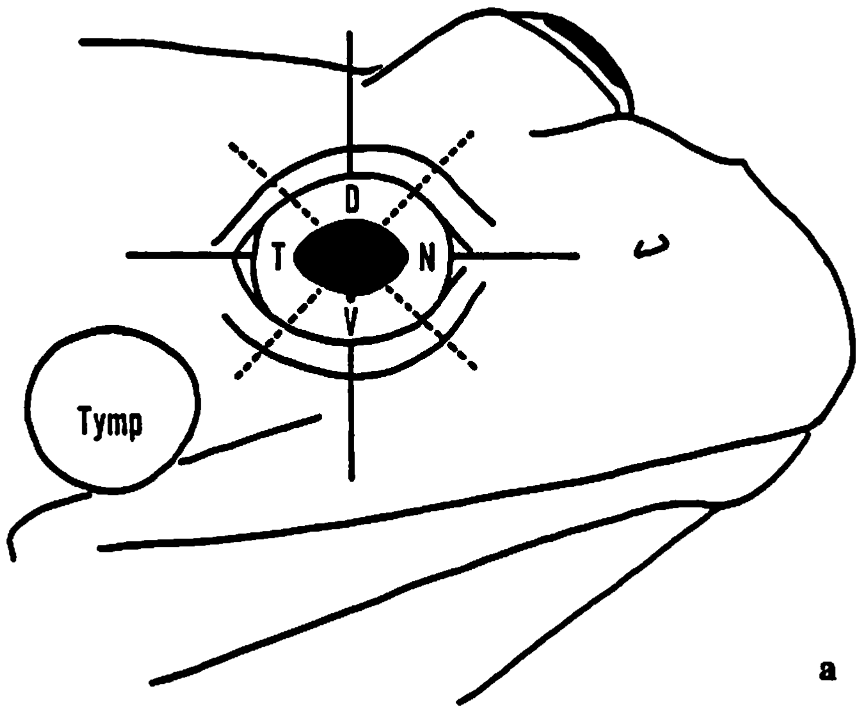
Although the optic nerve projection sites in thalamus are called "nuclei" they would more accurately be labeled "retinal recipient neuropil." They are actually regions of dense synaptic contact with the optic nerve. Thalamic cell bodies send dendrites into the nuclei to contact retinal ganglion cell axons. Few axons of thalamic neurons extend into this region [Scalia & Fite, 1974].

Optic nerve terminations within the tectum are also far from the cell bodies of the neurons with which they form synapses [Grüsser & Grüsser-Cornehls, 1976]. These terminals all lie in the relatively cell-free superficial layers of the tectum. Apical dendrites from cells in deeper tectal layers project into the superficial layers.

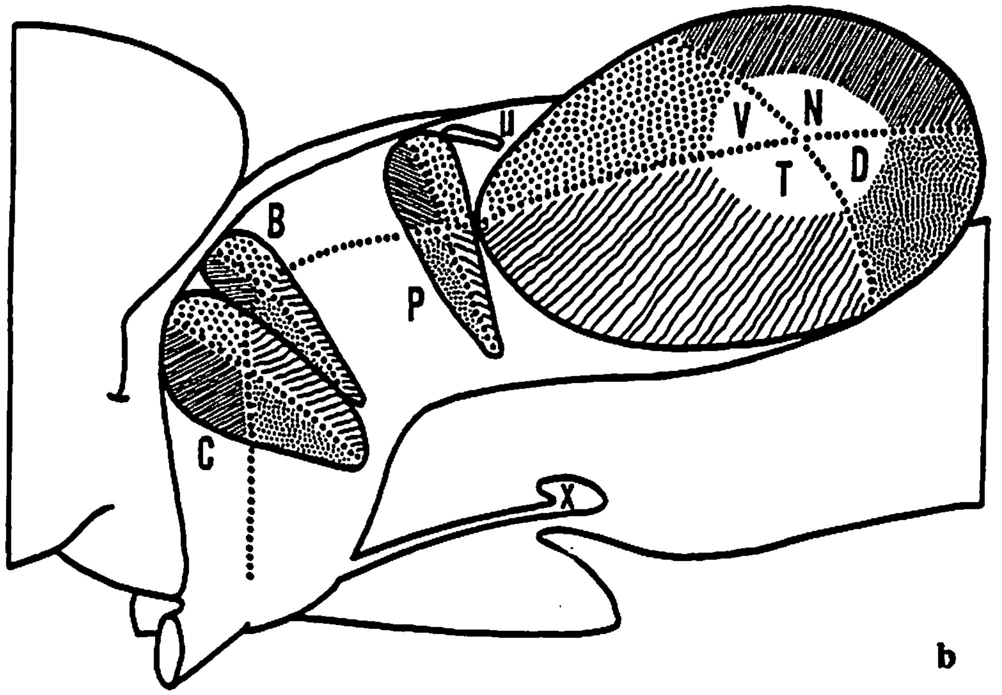
Fig. 7 - Optic Nerve Projection Sites.

(a) Diagram showing the labeling convention for the four quadrants of the eye: temporal T (anterior visual field), dorsal D (inferior visual field), nasal N (posterior visual field), and ventral V (superior visual field).

(b) Flattened sketch showing the left side of the brain and the major optic nerve projection sites. C is the corpus geniculatum, B the nucleus of Bellonci, P the posterior thalamic nucleus, U the uncinate pretectal nucleus, and X the basal optic nucleus. The large egg-shaped structure to the upper right is the optic tectum. Quadrant labeling is appropriate for projections from the contralateral (right) retina. The corpus geniculatum, nucleus of Bellonci, posterior thalamic nucleus, and the uncinate pretectal nucleus also receive ipsilateral projections. Note the mirroring of tectal projections in the thalamic nuclei. Reprinted by permission from Scalia and Fite [1974].



a



b

Here they intertwine with afferent fibers to form a glomerular structure.

Nucleus isthmi - a source of binocular input to tectum

Besides direct monocular projections from the contralateral eye each hemisphere of the optic tectum receives an indirect projection from the ipsilateral eye. This ipsilateral projection is visuotopically organized and in register with the contralateral projection.[2] Thus, that portion of the tectum devoted to the binocular visual field receives a complete binocular projection. The indirect ipsilateral projection originates in the contralateral tectal hemisphere and is transmitted via the contralateral nucleus isthmi. Neural pathways responsible for this projection have been studied by Fite and Scalia [1976], Glasser and Ingle [1978], Gruberg and Udin [1978], Gruberg and Lettvin [1980], Wang et al. [1981], and Grobstein and Comer [1983]. Some of the efferent fibers from the tectum project to the ipsilateral nucleus isthmi. This nucleus, in turn, sends its fibers to both the ipsilateral and the contralateral tecta. The resulting connections yield the visuotopic mappings shown in Fig. 8.

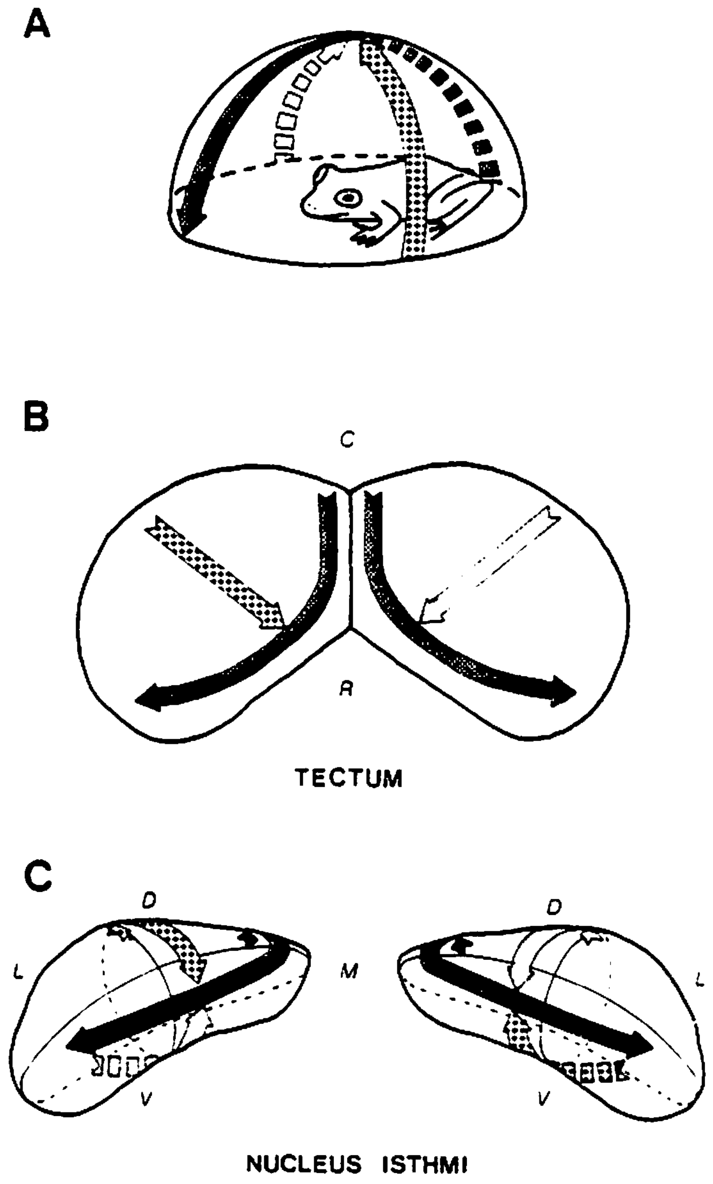
2 The notion of two retinotopic projections being in register implies the existence of a surface in space, any point on which maps through both eyes to the same point on the neural surface. In animals exhibiting vergence, this surface is called the horopter and, being the surface of zero disparity, always contains the fixation point. Frogs and toads do not exhibit vergence but nevertheless there is such a surface. In *Rana esculenta* the horopter passes through the eyes, is circular in cross section, and has a diameter of from 5 to 10 cm. in the longitudinal plane. [Gaillard & Galand, 1980].

Fig. 8 - Retino-Tectal and Isthmio-Tectal Projections.

(a) The upper visual field is represented by arrows. These arrows are used in the other illustrations to schematically represent projections of the visual field in neural structures.

(b) This dorsal view of the tectum (caudal end upward) shows the projection of the visual field formed by terminals of retinal ganglion cells.

(c) The nuclei isthmi are represented as shells in a view facing the rostral pole. M, L, D, and V indicate medial, lateral, dorsal and ventral positions. The locations in the contralateral tectum to which cells on the surface of the nuclei project are shown by the corresponding arrows. Reprinted by permission from Gruberg and Udin [1978].


A
B
C
R
TECTUM
C
D
L
M
V
D
L
V
NUCLEUS ISTHMI

Binocular units in tectum and thalamus

Electrode recordings have been made from binocularly active neuronal units in both tectum and posterior thalamus. The discovery of units that respond to stimulation from either eye shows that tectum and posterior thalamus are not only sites for the reception of binocular stimulation but are also involved in the integration of binocular information.

Since frogs and toads neither exhibit vergence nor have a distinct foveal area, it is unlikely that their binocular depth resolution is based upon finely tuned disparity sensitive units. If two eyes converge to fixate the same small region of visual space on the foveas of their two retinas then the range of disparities to be considered during binocular matching will be small and centered about zero. This range is known as Panum's fusional area [Panum, 1858]. Without vergence the range of disparities to be considered must cover not just those within a narrow fusional area but must extend over a large portion of the retinal surface. Electrode recordings within both tectum and thalamus confirm this supposition. Fite [1969], and later Skarf and Jacobson [1974], recorded from binocular cells in deep tectal layers. These cells have large receptive fields (mean: 76°), and tend to be multi-modal, responding to vibratory, tactile, and auditory stimulation as well as visual stimulation. Brown and Marks [1977], recording in thalamus, demonstrated the existence of similar multi-modal binocularly-sensitive units having even larger receptive fields. Raybourn [1975], and Finch and Collett [1983] report other more narrowly tuned binocular units in tectum. Both the tectal and thalamic binocular units exhibit spike frequencies that indicate a summation of contralateral and ipsilateral stimulation, and thus they are responsive over a broad range of disparities. This is in contrast to the excitatory/inhibitory interactions found in mammalian disparity-tuned units [G. Poggio, 1979].

Efferent pathways from tectum and thalamus

Tectal efferent pathways ascend into the thalamus and descend to both the nucleus isthmi and the brain-stem motor nuclei. In this study we will be most interested in the descending pathways. Rubinson [1968] used fiber degeneration techniques to trace the anatomy of these pathways in *Rana pipiens*. He found two distinct descending paths. One path extends ipsilaterally to the lateral brainstem tegmentum and the nucleus isthmi. The other path projects contralaterally to the medial brainstem tegmentum and spinal cord.[3]

Two major efferent pathways also project from the thalamo-pretectal region to downstream brain regions. Ingle [1983] used horseradish peroxidase (HRP) staining in bilaterally-tectal-ablated frogs to trace these two pathways from the pretectum into the medulla. He found that an ipsilateral descending route converges with the tectobulbar route through the medulla. A second route crosses the midline at the ventrolateral boundary of the tegmentum and then runs down the ventromedial wall. Later, by HRP backfilling, Ingle determined that a dense posterior medial cell group in pretectum contributes to both the medial (crossed) and lateral (ipsilateral) paths, and that a more lateral pretectal cell group contributes only to the ipsilateral path.

[3] See also Corvaja [1981].

The pathways that descend to the brain-stem from both the thalamo-pretectal region and tectum share a certain symmetry. In both cases there are ipsilateral as well as crossed pathways. Cell layers closest to optic nerve terminals contribute mainly to the ipsilateral pathways. Denser cell layers, farther from the terminals, contribute to both ipsilateral and crossed pathways. Fig. 9 (from Ingle [1983]), depicts these pathways. Besides showing the crossed vs. ipsilateral symmetry of the pathways this figure indicates that the tectal crossed pathway passes the midline at the ventral surface of the tegmentum. The pretectal pathway crosses the midline caudal to the tectum in the area of the isthmus.

Functional Analysis of the Major Visuo-Motor Centers

Retina

Besides its obvious function as the visual receptor surface of the brain the retina of frogs and toads plays a role in pattern recognition. Pattern selectivity inherent in the retina was investigated in the frog earlier than in any other species. In their pioneering study Lettvin et al. [1959] recorded from both the optic nerve and the tectum of frogs, *Rana pipiens*, while moving objects on the inside of a hemisphere which had been aligned with the optical axis of one eye. In this way they were able to identify four distinct classes of retinal ganglion cell. Type 1, *sustained contrast detectors*, have the smallest receptive fields (nominally 2 degrees) and respond vigorously to a light-dark edge that is moved into the receptive field and then held stationary. Type 2, *net convexity detectors*, have larger receptive fields (7 degrees) and respond particularly well to small objects moving with respect to the

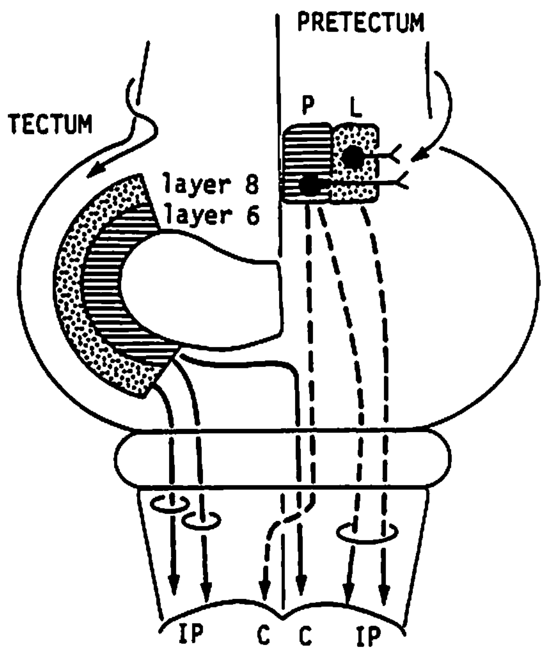


Fig. 9 - Tectal and Thalamo-Pretectal Efferent Pathways.

Solid lines indicate descending motor pathways from tectum, dashed lines from thalamus-pretectum (TPT). The most superficial tectal efferent layers (8 and 7) project ipsilaterally (IP). Deeper layers (6) contribute to both ipsilateral and crossed (C) pathways. Crossed tectal pathways pass the midline in the isthmial region. Likewise, projections from TPT project ipsilaterally from sites near to optic nerve terminals (L) with more distant somas (P) sending axons along both ipsilateral and crossed paths. Crossed TPT fibers do not pass the midline until they are just caudal of the tegmentum. Reprinted by permission from Ingle [1983].

---

background. Type 3, *moving-edge detectors*, respond to an edge moving through their 12-degree receptive fields. Type 4, *net dimming detectors*, have the largest receptive fields (about 15 degrees). Cells of this type respond to large dark objects entering the receptive field or to a rapid reduction in general lighting level. The discovery of these pattern selective units demonstrates that the retina must be considered as a pattern discriminating device and not merely a light-gathering device.

Because of retinal pattern selectivity it is unlikely that visual stimulation in the brains of frogs and toads consists of a fine-grained pattern that encodes retinal illumination level. It is more likely that certain visual stimuli are emphasized and other stimuli suppressed so that the total stimulation pattern is rather sparse. Although Fischer [1973] has shown that it is possible to reconstruct a fine-grained image from pattern-selective input in cats it is a matter of conjecture whether or not frogs and toads utilize such a reconstructive process.

It is an oversimplification to state that the retina is capable of discriminating, for instance, prey from barrier. However, this sort of sensory and functional differentiation is observable in tectum and thalamus.

Functional differentiation between thalamus and tectum

There is extensive evidence associating the tectum with prey detection and the thalamus with barrier detection. For instance, electro-physiological studies show that tectal visual units all seem to be motion sensitive [Grüsser & Grüsser-Cornehls, 1976], but there are at least two classes of thalamic units that are sensitive to stationary stimuli. These are the blue-sensitive *on* units reported by Muntz [1962], and stationary edge detectors reported by Ewert [1971], Brown and Marks [1977], and Ingle [1980]. This difference between tectal and thalamic units suggests that these two areas may be responsible for the processing of very different visual stimuli. In particular, they implicate tectum in prey detection and thalamus in barrier detection. Lesion studies confirm this supposition.

Evidence found by Ingle [1977] during an optic nerve regeneration study strongly implicates the tectum in prey recognition. In his experiment Ingle performed unilateral tectal ablations in frogs and then allowed time for the optic nerve to regenerate to the opposite tectal surface. Frogs "rewired" in this way would respond to prey dummies presented in the visual field contralateral to the ablation as if they were in the opposite field. Their wrong-way turning and snapping was calibrated with the position of the prey in the visual field – the more eccentric the target the greater the amplitude of the misdirected turn.

The behavior of Ingle's "rewired" animals in the presence of barriers strengthens the argument that barrier navigation is not tectally mediated. These animals showed strong dissociation of function between orientation to prey and navigation around barriers. With a barrier placed in the unaffected visual field and a prey in the affected field, these frogs initiated turns around the barrier as if the prey were behind the barrier. With the sides of prey and barrier reversed the same animals made normal orientation turns toward the prey, ignoring the barrier in the opposite visual field.

In experiments with atectal frogs, Ingle [1982] was again able to demonstrate the division of function between thalamus and tectum. Bilateral tectal ablation abolished both orienting and prey catching responses. However, the same animals were still completely reliable in clearing barriers to avoid noxious tactile stimulation. Their behavior was such as to just clear the barrier edge, whereas intact animals invariably turned away from the threat.

Evidence for depth perception in thalamus

Ingle [1982] also showed that his atectal frogs were able to discriminate the depth of barrier surfaces during avoidance behavior. Fig. 10 shows the experimental design which he used to demonstrate this. An opaque barrier placed around the frog's head had one opening, through which could be viewed a series of vertical bars placed on a surface several inches behind the enclosure. Near this opening was a similar configuration of vertical bars, but these bars were placed on the surface of the enclosure. The bars in this second set were calibrated so that their size and spacing would appear to be identical to the bars visible through the opening. During avoidance trials the frogs preferred the real opening as an escape route, rarely orienting towards the false opening. This experiment clearly showed that frogs can discriminate the depth of surfaces without the use of tectum. Since it receives binocular innervation the thalamus is implicated in this discrimination.

Motor pathways from tectum and thalamus

Studies of the pathways descending from both tectum and thalamus not only confirm the thesis segregating prey detection to tectum and barrier detection to thalamus but also give some indication of the types of motor activity controlled by these two areas. Studies by both Ingle [1983] and Grobstein et al. [1983] showed that the crossed and uncrossed tracts descending from both the thalamo-pretectal area and the tectum perform distinctly different functions.

The crossed pathways from tectum have been implicated in the orienting response. Ingle [1983] investigated the effects of severing the crossed descending pathways from tectum by lesioning the ansulate commissure at the ventral surface of

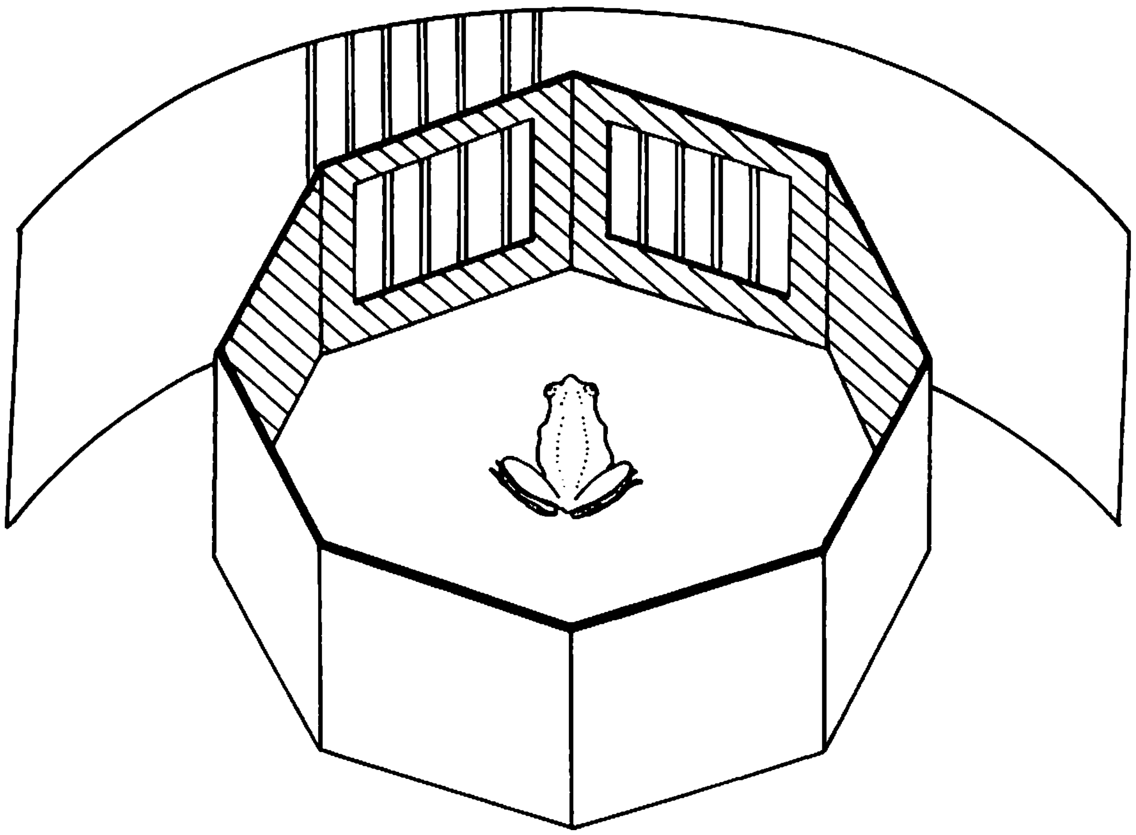


Fig. 10 - Depth Vision in Atectal Frogs.

A barrier placed around a frog had one panel through which was cut an opening. A grid of vertical stripes affixed to a second barrier several inches behind the first one could be viewed through the opening. Another panel had a false opening which consisted of a grid of vertical stripes placed directly on the barrier's surface. The size of the false opening was the same as the real opening. The width and spacing of the stripes on the false opening were scaled to match those perceived through the real opening. Even after total tectal ablation, frogs would aim for the true opening to escape noxious stimulation. Reprinted by permission from Ingle [1983].

the tegmentum. Frogs prepared in this way were highly motivated to strike at prey but could not orient towards their prey. Snaps were always straight ahead. For more distant prey they hopped forward, again without the usual orienting response. However, for prey positioned above the head they were able to make the appropriate vertical movements.

If the crossed descending tract from the pretectal area is severed, frogs lose their ability to sidestep around barriers. Ingle transected the crossed fibers from pretectum by splitting the isthmus. These frogs retained their ability to turn towards prey. However, they were unable to turn or sidestep to negotiate barriers.

Frogs with severed crossed pretecto-fugal pathways show a motor but not a sensory deficit. For example, although a tail pinch would send them crashing into a barrier if it crossed the midline by more than 15 degrees, the same animals were able to round a barrier to catch prey if the barrier edge was on the midline, so that a sidestepping movement was not required. Also, the lesioned animals would consistently suppress strikes toward prey located behind a frontal barrier.

The evidence from his lesion experiments led Ingle to make the following functional description of the crossed and uncrossed tecto- and pretecto-fugal pathways. First, the uncrossed (ipsilateral) pathways are involved in symmetric motor behavior, i.e. snapping or walking straight ahead or moving the head and body vertically. Thus, the ipsilateral tectal pathways govern strike vs. no-strike activity and the ipsilateral pretectal pathways govern whether or not to avoid a barrier surface. The crossed pathways, on the other hand, govern one-sided or asymmetrical movements. For tectum they direct turning or orienting movements towards prey, and for pretectum they govern sidestepping or turning around barriers.

Depth perception dissociated from orientation

Ingle's experiments with split-tegmentum frogs also showed that these animals retain their ability both to estimate depth and to use this depth information to regulate their behavior. Frogs will snap at a prey if it is within a frontal "snap zone," but prefer to hop towards a prey if it is located beyond this zone [Ingle, 1982]. Ingle's split tegmentum frogs would not turn towards prey; all snaps were straight ahead. However, even though they could not aim correctly these frogs would still snap when a prey was located anywhere within the snap zone and hop forward when a prey was located outside of the snap zone.

Motor activity and tectal retinotopy

Early evidence indicated that the tectum directed motor output and that it did this in a retinotopically organized way. By means of electrode stimulation in tectum Ewert [1971] was able to elicit typical prey catching responses that were consistently directed at the spatial location predicted from the organization of the retinotectal map.

However, Grobstein et al. [1983] found that the motor pattern generators underlying the prey catching behavior of frogs do not reside in tectum. They found that although tectal lesions abolish visually elicited orientation responses, orientation to tactile stimuli is preserved. Conversely, lesions to *lateral torus semicirculus* abolish tactilely induced orientation while sparing visually induced orientation. Thus, they concluded that the motor pattern generators must reside in a brain region that receives input from both tectum and torus.

In the same series of experiments Grobstein et al. also found that although visual input to tectum is retinotopically organized this retinotopy is not preserved in the tectal output. Small lesions which cut across part of the tecto-motor neuraxis did not produce the expected orientation scotomas. Instead, frogs with these lesions tended to undershoot the stimulus position when orienting to a stimulus contralateral to the lesion. However, even though they lost accuracy the lesioned animals continued to be able to initiate orientation turns to all angular directions. It was not until the neuraxis was completely severed that orienting responses contralateral to the lesion were abolished. If a retinotopically organized map from tectum to motor output did exist, the lesions to the neuraxis would have had distinct local effects throughout the affected hemifield. Instead, the effect appeared to be distributed, with the number of intact fibers governing accuracy over the entire region.

The idea of a retinotopically organized map from tectum to motor output was even more seriously challenged by a second experiment [Grobstein et al., 1983]. In this experiment both a unilateral tectal lobe removal and a hemisection of the neuraxis were performed. Thus, when the tectal removal and hemisection were done on the right side, tectal activity from the left eye's visual field was destroyed and the crossed descending pathway from the left tectal lobe was also severed. The only descending tectal pathway remaining intact was the uncrossed ipsilateral pathway on the left side (refer to Fig. 9). As would be expected, these animals could not make visually elicited turns to the side ipsilateral to the lesions. However, they were able to make turns to the side contralateral to the lesions but only in response to an ipsilaterally located stimulus. The amplitude of the wrong-way turn could be increased by placing the stimulus at a more eccentric position in the opposite visual field. The fact that these unitectal animals could elicit turns into the visual field covered only by the ablated tectal lobe makes it highly unlikely that each tectal locus can be associated with a turn of a particular direction.

The argument that orientation turning cannot be topographically organized can also be made from considerations of visuo-motor geometry. Since the body turning axis is offset from the origin of an eye-centered coordinate system, various orientation responses are appropriate for a single retinal location. Fig. 11 from Grobstein et al. [1983] demonstrates this point. Objects A, B, and C all stimulate the same retinal locus and thus the same tectal position. However, with the body turning axis placed on the midline but behind the plane of the pupils, A results in a right turn, B in no turn, and C in a left turn. This ambiguity in relating turn direction to tectal topography can only be resolved by considering target depth as well as angular position.

Summary of the frog/toad visual system

In our discussion of the visual system of frogs and toads we have made the following points:

1) frogs and toads make use of depth vision to locate both prey and barriers,

2) they utilize both binocular and monocular depth cues,

3) monocular depth cues are from lens accommodation,

4) there is a substantial region of binocular overlap in the visual field,

5) there are retinotopically organized visual maps in both thalamus and tectum,

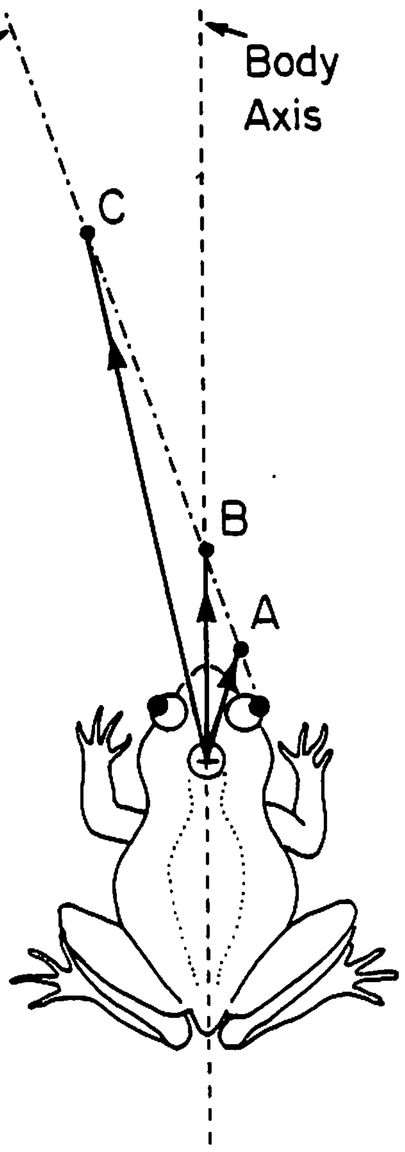


Fig. 11 - Orientation Must be Spatially Directed.

Various orientation responses (arrows) are possible for a single retino-tectal location. This is due to the dislocation of the body-turning axis (origin of arrows) from an eye-centered coordinate frame. Points A, B, and C all stimulate the same retinal position but the appropriate orientation turn for the frog is a right turn for A, no turn for B, and a left turn for C. The conclusion is that orientation turning must be spatially directed and not directed by retinal position alone. Redrawn by permission from Grobstein et al. [1983].

6) the visual maps cannot be assumed to represent local illumination level, since the visual input to the brain is feature-encoded,

7) both tectum and thalamus receive binocular visual input and both have binocularly activated neuronal units,

8) tectal binocular input is via an indirect relay in nucleus isthmi but thalamic binocular input is direct,

9) tectum is implicated in the recognition and localization of prey and thalamus is implicated in the detection and localization of barrier surfaces,

10) retinotopy may not be preserved in the efferent pathways from tectum,

11) orientation turning depends upon the depth of a target as well as the position to which it projects in tectum.

Modeling Depth Perception in Frogs and Toads

Four assumptions made in the depth models discussed earlier are inconsistent with the data on the visual system of frogs and toads. Three of these assumptions have to do with the nature of the images available to the depth resolving system. These are assumptions of 1) vergence, 2) static images, and 3) dense images. The fourth assumption is more fundamental. It is the assumption that 4) the model's purpose is to produce a complete depth-mapping of the image surface.

All of the depth models are based on the assumption that the image pair to be processed has been obtained from an imaging system with the capability of vergence. Due to vergence the disparities in the image will be small and centered about zero. Thus, the matching algorithm can be constrained to consider only those matches that lie within a narrow disparity range and the smaller disparities can be weighted more heavily than the larger ones. Since frogs and toads do not demonstrate vergence this is an inappropriate assumption for a model of their process of depth perception.

Since the models operate on a static image pair, the only available depth cues are those that can be obtained directly from these images. Any active process that a functioning animal might utilize to explore its visual surroundings cannot be considered. This restriction explicitly rules out treatment of depth cues that could come from motion parallax due to eye or head movements, change in image size during approach, or change in image focus due to lens accommodation. It is known that frogs and toads use lens accommodation as part of their depth perception process. Thus, at least this active process must be incorporated in the model.

The models also depend upon there being intensity variations between neighboring picture elements across most of the image surface. For example, most of the models were tested using either random-dot stereograms or digitized photographs. It is not clear that any of the proposed mechanisms would be appropriate for producing a depth map from the more sparse feature-encoded visuotopic maps available to frogs and toads.

Finally, all of the previous depth models were designed to build a visuotopically organized depth map. None of these models consider the end use of the depth information which they produce. The intent in developing these models was to provide a general solution to the binocular depth resolution problem and not to design an algorithm that could be integrated into the activity pattern of a specific animal. In our study of depth perception we wish to define algorithms which, even at the expense of generality, are consistent with both the known behavior and neural substrates of frogs and toads.

A recent series of experiments by Collett and Udin [1983] challenged the idea that a depth map, produced in tectum, supplies the depth information necessary for prey catching. They were able to demonstrate that toads with large electrolytic lesions to nucleus isthmi, and thus deprived of binocular input to tectum, were still able to use binocular cues to judge prey depth. Collett and Udin's lesioned animals were as accurate as controls in snapping at a single prey, and when base-out prisms were placed in front of their eyes the lesioned toads made the undershooting errors predicted for binocular depth estimation.

We have defined some of the constraints that guided our efforts at modeling depth perception in frogs and toads. In the remaining chapters we develop two quite different models guided, more or less, by these constraints. Finally, we contrast the models and show how the actual depth perception process might make use of elements of both models. The first of these models does make use of the concept of a general depth map. It falls naturally in the family of models discussed at the beginning of this chapter. It uses cooperative computation to build its depth maps using both binocular disparity matching and lens accommodation as depth cues. An important feature of this model is that it can work both with and without binocular depth cues. The second model avoids the use of a depth map and instead provides the spatial location of a single point in the visual field. This model is much more specific about the interaction of binocular and lens accommodation mechanisms. It represents this interaction as part of an action-oriented feedback control loop which is specialized to prey capture.

Conclusions

Our comparison of the anatomy, physiology, and behavior of frogs and toads with a series of recent models of binocular depth perception has shown that none of these models can adequately represent the depth perception process in frogs and toads. We have noted that a successful model 1) must not depend upon vergence; 2) must be able to account for the use of active processes, particularly lens accommodation; 3) must be able to operate on sparse images; and 4) must not depend upon a tectal depth map. Our discussion of the visual system of these animals argues that 1) the process used to estimate the depth of barrier objects might be very different from that used to estimate the depth of prey, 2) prey depth discrimination is tectally mediated whereas barrier depth is determined in thalamus, and 3) both binocular and monocular (from lens accommodation) depth cues are used in prey depth determination.

CHAPTER III

INTERACTION OF ACCOMMODATION AND DISPARITY CUES

Summary

In this chapter we introduce a model of depth perception that explores one way in which lens accommodation cues can be used to help disambiguate the correspondence problem of stereopsis. The model conforms to the constraints and assumptions of previous cooperative stereo models but extends their computational strategy. It consists of two cooperative depth discrimination processes, each acting to build a depth map, with one process receiving monocular depth cues based on accommodation and the other receiving binocular depth cues based on disparity. Mutual excitatory connections between the maps allow the model to converge to a single solution, whose accuracy is governed by binocularity. The model will also function in a purely monocular mode if binocular input is removed.

Although the model is of general interest as an extension to the class of cooperative stereo models, it was originally developed to explain how frogs and toads can do binocular depth perception without the use of cues from vergence. Neurophysiological data on the visual system of frogs and toads are used to constrain the choices available in constructing the model, and results obtained from simulation runs are compared with data from behavioral experiments with these animals.

Introduction

The problem of the computation of a depth-map from the binocular correspondence of disparate retinal images is complicated by the potential for ambiguity in the process of pairing local features. Fig. 12 shows the simplest example of this ambiguity. Because of ambiguity a system for stereoscopic depth-perception must have sources of disambiguating information.

In Chapter 2 we showed that most sources for disambiguating cues proposed in previous models of stereoscopic depth perception come either explicitly or implicitly from the assumption of binocular convergence. For example, in their cooperative stereo algorithm, Marr and T. Poggio [1976] bias the selection process by considering only matches that lie within a narrow (6 pixel) range of horizontal disparity before resorting to cooperativity between nearby points. Hirai and Fukushima [1978] follow a similar scheme but make an even more explicit assumption of vergence. Their model weights matches in a way which gives preference to those with the smallest disparity. In addition, it incorporates an ordering rule, clearly dependent upon vergence for correctness, that enforces a global consistency among matched points. More recent models, employing spatial-frequency tuned channels to assist the matching process [Marr & T. Poggio, 1979; Frisby & Mayhew, 1980], continue to rely on the assumption of vergence.

Unlike these previous studies, which take the mamallian visual system as a guide, we use the visual system of frogs and toads as our point-of-reference. Because frogs and toads do not move their eyes to fixate prey [Ewert, 1980] the range of binocular disparities that they must consider cannot be confined to a

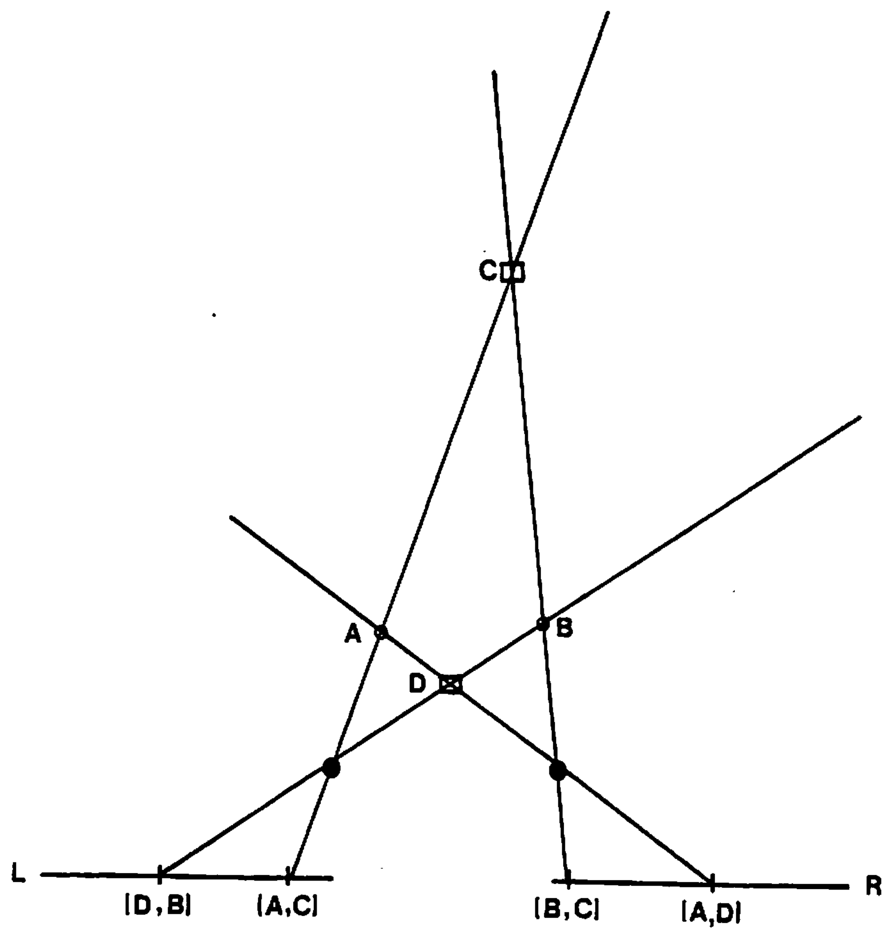


Fig. 12 - Ambiguity in Binocular Image Matching.

L and R represent left and right retinal surfaces, filled circles are pupils, lines indicate lines-of-sight. The pair of spatial points (A, B) cannot be distinguished from the pair (C, D) of "ghost" targets by simply considering the positions of stimulation on the two retinal surfaces.

narrow fusional area [Panum, 1858] but must extend over a large portion of the image surface. Consistent with this fact, binocular visual units in both the thalamus and tectum of frogs and toads appear to sum ipsilateral and contralateral input over broad receptive fields [Fite, 1969; Raybourn, 1975; Brown & Marks, 1977]. This is in contrast to the excitatory/inhibitory interactions found in narrowly-tuned mammalian disparity-selective units [G. Poggio, 1979].

Frogs and toads are able to estimate the depth of prey using either binocular or monocular depth cues. Collett [1977], in a series of behavioral experiments in which he used lenses and prisms to dissociate monocular from binocular cues, showed that toads (*Bufo marinus*) strongly favor binocular cues when they are available. Snapping distance, from his data, is closely matched by a formula that gives a 94% weighting to binocular cues and only 6% to monocular cues. The picture is complicated, however, by the fact that, despite their preference for binocularity, both frogs and toads continue to snap accurately at prey even after monocular blinding [Ingle, 1976; Collett, 1977]. The monocular depth cues used to accomplish this come, almost certainly, from lens accommodation [Collett, 1977; Jordan et al., 1980].

The neural mechanisms underlying the binocular/monocular depth resolving system of frogs and toads probably reside in both the optic tectum and the thalamus. Both the tectum and several thalamic nuclei receive binocular visual stimulation [Scalia & Fite, 1974]. Optic nerve projections to tectum are monocular but information from the ipsilateral eye is available from a cross-tectal relay via the nucleus isthmi [Fite & Scalia, 1976; Gruberg & Udin, 1978; Gruberg & Lettvin, 1980;

Grobstein & Comer, 1983]. Binocular input to thalamus is direct. The thalamic nuclei appear to be specialized for the detection of barriers and the tectum for prey. Both of these brain centers have been implicated in visually-guided behavior involving the use of depth information [Ewert, 1971; Ingle, 1977].

Besides their reliance upon vergence cues, earlier cooperative stereoscopic depth models have only been tested against visually-rich stimulation such as is found in random-dot stereograms or digitized photographs [Dev, 1975; Julesz, 1971; Marr & T. Poggio 1976; Trehub, 1978; Marr & T. Poggio, 1979; Frisby & Mayhew 1980]. The visual stimulation reaching the brain centers of frogs and toads may be much more sparse, due to the presence of highly specialized retinal pattern selective units [Lettvin et al., 1959].

Design of the Model

In the development of our model we wished both to adhere to the general methodology employed in earlier cooperative stereo models and to address the unique features of the depth resolving system of frogs and toads. Our goals were 1) to eliminate dependency on vergence by using monocular cues from lens accommodation to help disambiguate binocular cues from image matching; and 2) to define a mechanism in which binocular depth cues dominate but that can still function when only monocular depth cues are available.

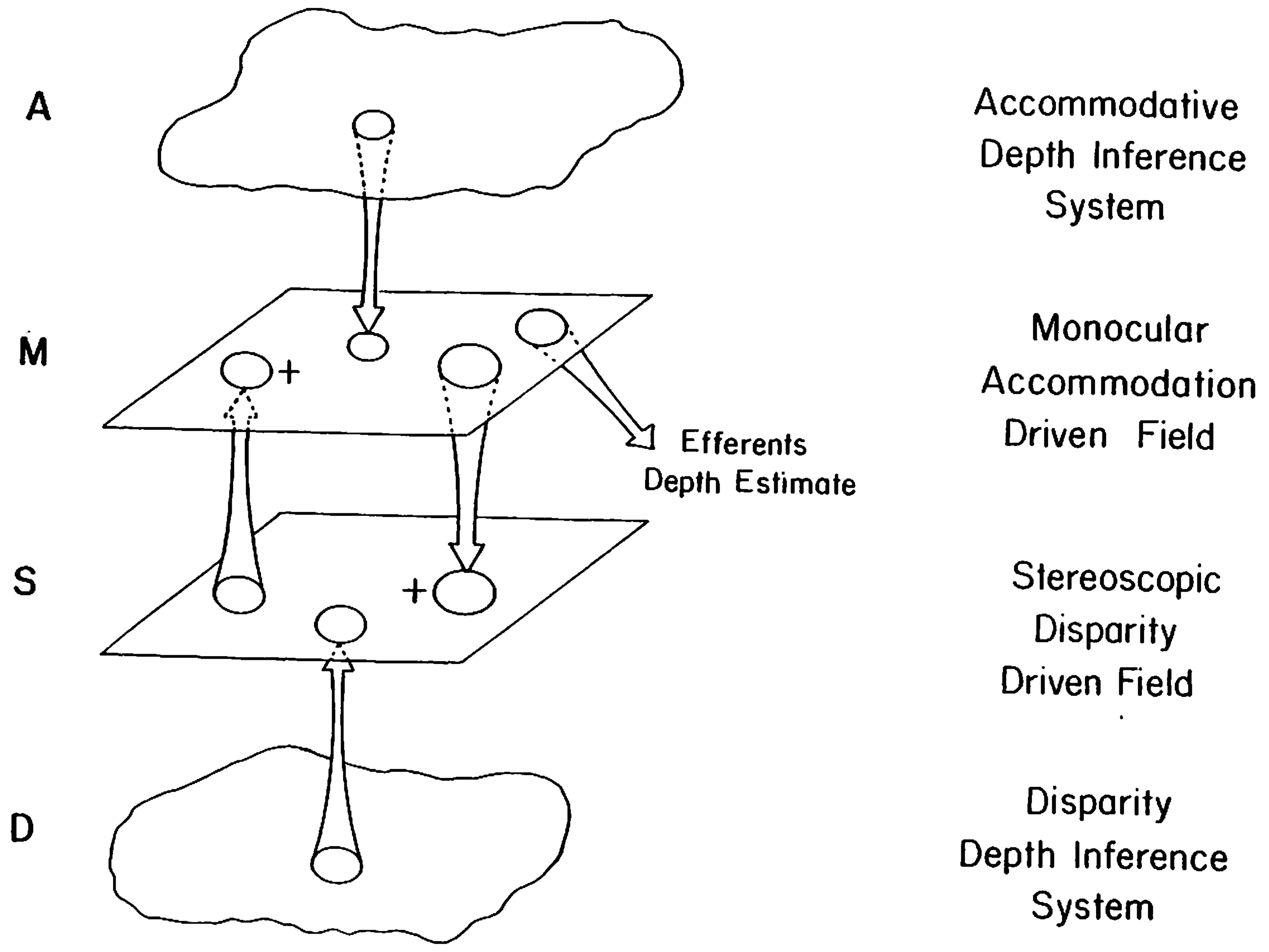
Structure

In place of a single depth-resolving process utilizing only binocular depth cues, our model employs a pair of interacting processes, each with its own source of depth cues. One process receives monocular depth cues from lens accommodation, and the other receives binocular depth cues based on disparity matching. The two processes cooperate to augment each other. Fig. 13 depicts the overall connectivity of the model. It shows binocular and monocular depth inference systems supplying input to two fields. Each of these fields represents a cooperative depth inference process operating over a visuotopically-organized spatial map. Arrows drawn between these two fields represent facilitatory interconnections between the maps that connect a region on one map with the spatially-corresponding region on the other map. Because of these interconnections, a local build-up of excitation in one of the maps provides additional excitation to the same region in the other map. Thus, the process receiving monocular input and the process receiving binocular input act to bias each other. Since the connections between the processes are reciprocal, similar monocular and binocular depth cues are strongly mutually reinforcing.

In this system, monocular cues from lens accommodation need not be used to estimate depth directly. Binocular depth cues are ambiguous simply because two or more cues at the same retinal position can have identical weights. Accommodative cues may be quite weak, compared with the binocular cues, and still provide enough of a bias to resolve the binocular ambiguity. If the model is tuned in this way, accommodative cues need not be acute or even very accurate for the scheme to work properly. Since accommodative cues simply serve to augment corresponding

Fig. 13 - Connectivity of the Complete Cooperative Depth System.

The various layers of the model and their interconnections are shown. Layer A represents an inference system that provides monocular depth cues from lens accommodation, and layer D represents an inference system that provides binocular cues from disparity matching. Layers M and S represent spatially organized fields over which two depth-mapping processes operate. Arrows from layers A and D to these fields indicate that field M receives only monocular depth cues, and field S receives only binocular cues. The ovals and arrows between fields M and S indicate mutual excitatory interconnections that map each local region (oval) of one field onto the corresponding local region of the other field. By means of these interconnections, points of high excitation in one field provide additional excitation to corresponding points in the other field. Thus, there is a high degree of synergy wherever similar points are externally excited in both fields. Competition among depth estimates within each field assures that points excited in only one field will have little chance to sustain this excitation when there are other points receiving stimulation in both fields.


A
Accommodative Depth Inference System
M
Monocular Accommodation Driven Field
+
Efferents Depth Estimate
S
Stereoscopic Disparity Driven Field
+
D
Disparity Depth Inference System

binocular cues, the accuracy of the system will be dominated by the accuracy of the binocular estimates. However, since the accommodative cues are not ambiguous, they can be used to estimate depth, even in isolation from binocular cues.

Fig. 13 shows the efferent pathways from the model emanating from the monocularly driven map. This is meant to indicate that the monocularly driven map should be capable of directing motor coordination whether or not binocular information is available, even though binocular information dominates when it is available. In actual practice it makes little difference which of the two maps provides the efferents, since when the model is tuned for good performance the two maps tend towards nearly identical equilibrium states.

Constraints

In designing our model of depth perception we wished to adhere to the underlying principles of previous cooperative depth models but at the same time to incorporate the additional constraints derived from the visual system of frogs and toads. The prime constraints of most cooperative stereo models are 1) uniqueness: for each retinal position only one depth may be assigned, and 2) continuity: disparity (or depth) varies only gradually or not at all over most of the image [Dev, 1975; Marr & Poggio, 1976]. The constraints derived from the properties of the frog/toad visual system are 3) the ability to function both with and without binocular depth cues, 4) the lack of binocular depth cues from vergence together with the availability of monocular cues from lens accommodation, 5) sparsity of visual input due to the presence of pattern-sensitive mechanisms in the retina, and 6) the separation of visual input into distinct prey and barrier channels that project to two different brain regions.

All of the above constraints are mutually consistent, with the exception of continuity 2) and sparsity 5). Continuity implies the availability of dense images so that regions of depth discontinuity occur only at the edges of objects. Sparsity, however, implies that the image is mostly void with the exception of a few areas of select stimulation. This conflict was resolved by relaxing the continuity constraint to require only that most neighboring *stimulated* points share similar disparities.

Constraint 6), that the visual input is broken into separate prey and barrier channels, has three possible implications for depth perception in frogs and toads. The first is that the process of depth perception might take place before the functional differentiation occurs. This is deemed unlikely since atectal frogs are able to judge the depth of barrier surfaces during escape, but the same animals completely ignore prey objects [Ingle, 1982]. The other two possibilities are that identical depth perception processes are replicated in two different brain regions or that the process for estimating depth of prey is different from the process for estimating depth of barriers. For the remainder of this chapter we proceed as if two identical processes are used. This is done for theoretical reasons alone, and does not imply a belief that the two process are, in fact, identical. Treating both processes identically simply serves as a means for obtaining data that allow us to compare the functioning of the model on two very different kinds of visual input. In Chapter 5 we explore this matter further.

Coordinate systems

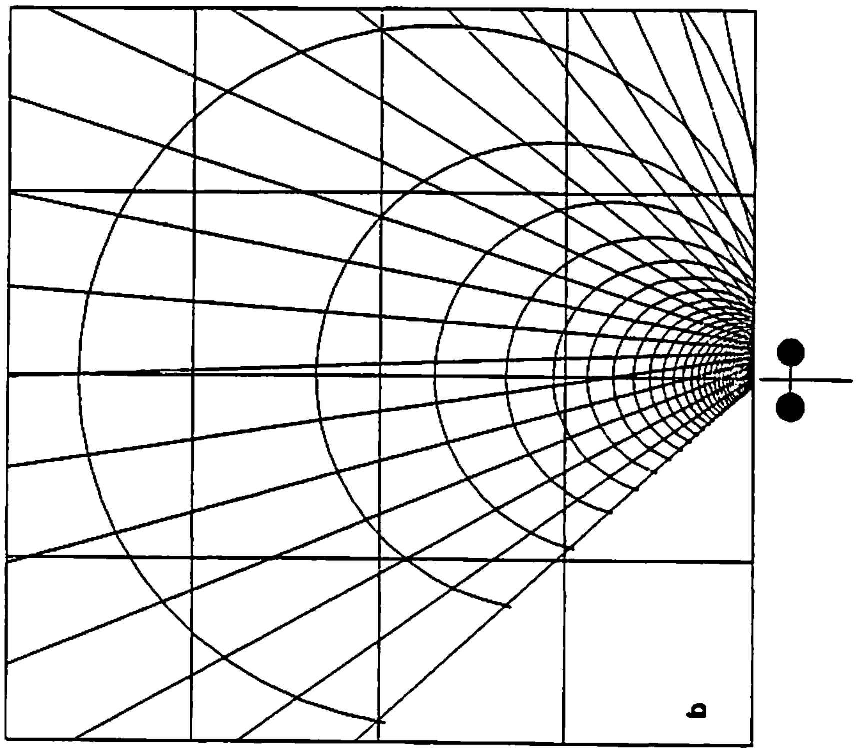
Although previous stereoscopic depth perception models have all employed a single cyclopean coordinate system, we chose eye-centered coordinate systems for the expression of our model. These coordinate systems plot angular disparity versus retinal angle separately for each eye. The choice of two eye-centered coordinate systems was made to provide a representation of visual space appropriate for treating both binocular and monocular depth cues. Since both the cyclopean and the eye-centered coordinate systems are based upon disparity they are equally appropriate for representing disparity-cued depth measurements. However, depth cues based upon accommodation are naturally eye-centered.

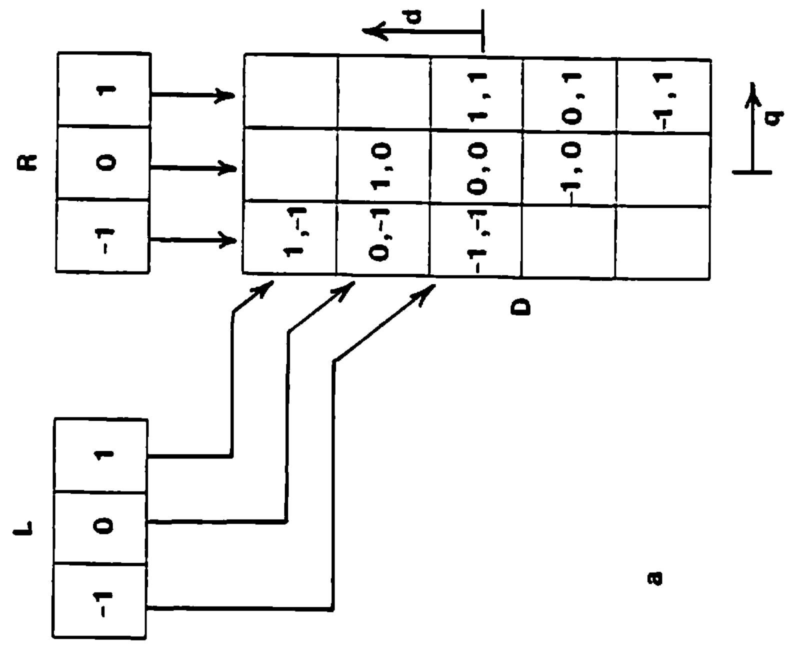
The problem was restricted to the consideration of one-dimensional retinas and a two-dimensional world, allowing these coordinate systems to be constructed as shown in Fig. 14a. For the right-eyed system the horizontal axis is defined by angular position on the right retina, so that a point on the right retina maps to a vertical line. The vertical axis measures angular displacement of the image on the left retina from the image appearing at the indicated position on the right retina, so that a point on the left retina maps to a diagonal. This displacement, or disparity, is taken as being positive when the portion of the image being considered on the left retina is to the right of that on the right retina. The left-eyed system is configured similarly but with the horizontal axis measuring position on the left retina. Fig. 14b depicts how the right-eyed system would appear if overlaid upon an

Fig. 14 - The Coordinate Systems of the Model.

(a) Construction of binocular depth estimates in retinal angle vs. disparity coordinates for the right-eye centered system. Array D represents a continuous plane in this coordinate system. The horizontal coordinate, q, represents position on the right retina and the vertical coordinate, d, represents angular disparity between the right and left retinal images. L is the left and R is the right retina. Number pairs within the cells of the array D indicate which points on the left and right retinas are represented by the cell of D in which they appear.

(b) Projection of the right-eyed coordinate system back onto an external Cartesian grid. Radial lines are at equal angular increments, arcs are lines of constant disparity spaced at equal increments of disparity. The spacing of the equi-disparity arcs illustrates the decrease of depth acuity with increasing distance from the animal. The object below the grid area is a schematic representation of a frog or toad with eye positions indicated by the large disks.


b



a

external Cartesian system.[1] The properties of optical depth resolving systems are reflected in the compression of the representation near to the eye, allowing fine depth resolution only for nearby objects.

Computational scheme

A general cooperative/competitive scheme, due to Amari and Arbib [1977], and built upon earlier work by Didday [1970] and Dev [1975], was used as a substrate for the model. Amari and Arbib consider situations where the input is a two-dimensional array, and the problem is to select a single distinguished element for each column of this array. Their solution consists of a homogeneous two-dimensional excitatory neural field interacting with a one-dimensional inhibitory neuron pool. Since this scheme selects a single element for each column of the input it incorporates the uniqueness constraint 1). The continuity constraint 2) is realized by mutual excitatory connections between neighboring columns of the field.

Fig. 15a shows how the Amari/Arbib scheme handles the depth-perception problem for the case of the projection of a two-dimensional scene onto one-dimensional retinas. The excitatory field is defined in the retinal angle (q) vs. disparity (d) coordinate system. The input to the field is a map that assigns a set of depth-likelihoods to each retinal position, and that is determined by a single depth-inference system. Regions of the field that are close in both retinal position and disparity are mutually excitatory, with the local spread of excitation fairly broad along the retinal position axis and quite narrow along the depth axis. Thus, similar

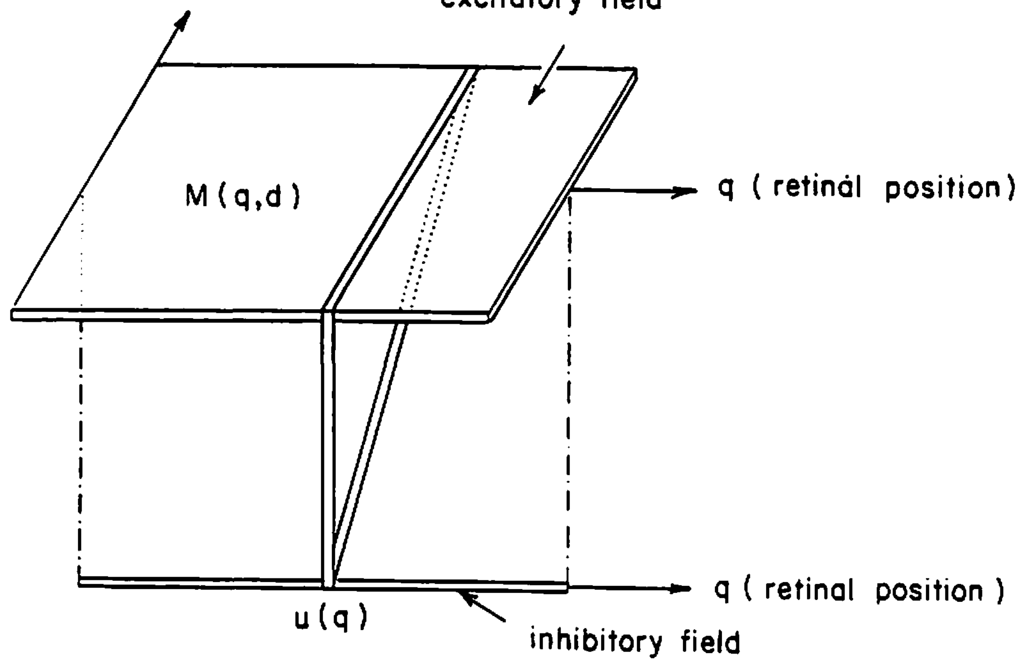
[1] See Appendix B for a description of the transformation from the internal retinal angle vs. disparity coordinate system to the external Cartesian system.

Fig. 15 - The Amari/Arbib Model.

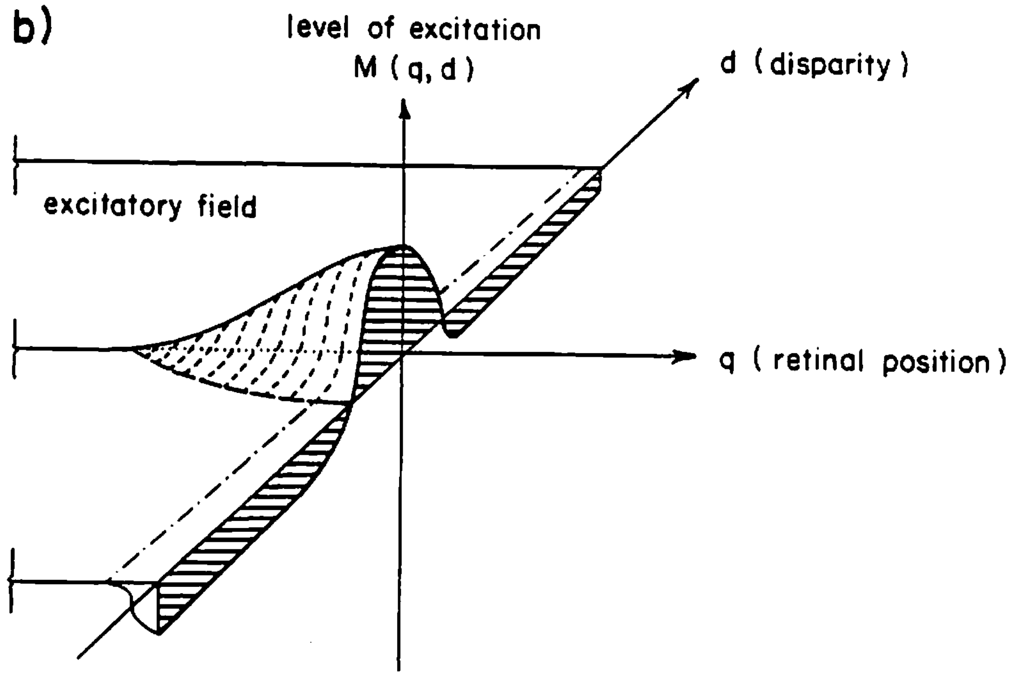
(a) This diagram is meant only to suggest the structure of the model and not the detailed connectivity. The labeling is appropriate for the solution of the depth-mapping problem for one dimensional retinas and one source of depth cues. Input to the excitatory field is a set of depth cues organized in a retinal angle vs. disparity coordinate system. The inhibitory field sums excitation for all disparities at each retinal position in the excitatory field and provides a reciprocal inhibitory feed-back. There is lateral facilitation in both fields, so that nearby elements within a field are mutually reinforcing. Redrawn by permission from Amari and Arbib [1977].

(b) This cut-away view shows the net spread of excitation and inhibition around a point-source of external stimulation in the excitatory field. It shows the sympathetic excitation of points in the field at nearby retinal positions and similar disparities. There is a spread of inhibition along the entire disparity axis, within a narrow band about the retinal position of the stimulus. Thus, other depth estimates at the same retinal position tend to be suppressed.

a)


d (disparity)
excitatory field
M (q,d)
q (retinal position)
q (retinal position)
u (q)
inhibitory field

b)


level of excitation
M (q, d)
d (disparity)
excitatory field
q (retinal position)

depth estimates at nearby retinal positions cooperate to reinforce each other. The portion of the inhibitory pool assigned to a particular retinal position receives an excitatory signal that is modulated by the sum of all excitation along a narrow band centered about that retinal position in the field. The inhibitory pool, in turn, sends inhibitory feedback to all elements at that retinal position. Thus, multiple depth estimates at the same retinal position must compete with each other in order to sustain field excitation. If parameters are properly chosen the field will reach an equilibrium state where excitation is maintained at only one depth region for each retinal position. Fig. 15b shows a cut-away view of the net effect on the field, due to both the excitatory and inhibitory connections, when a single point source of excitatory input is provided. The inhibitory effect upon points at the same retinal position but differing disparities and the facilitatory effect on neighboring points of similar disparities are apparent in this figure.

The Amari/Arbib model was originally developed as an analytical description of a general cooperative-competitive process, and can can be described by the pair of continuous non-linear integro-differential equations

$$\begin{aligned} T_m \dot{M}(q,d,t) &= -M(q,d,t) + \iint w_m(q-\zeta, d-\eta)\, f[M(\zeta,\eta,t)]\, d\zeta\, d\eta \; - \\ &\qquad K_m\, g[U(q,t)] + K_a\, A(q,d,t), \qquad (1) \\ T_u \dot{U}(q,t) &= -U(q,t) + \int w_u(q-\zeta)\, g[U(\zeta,t)]\, d\zeta + K_u \int f[M(q,\eta,t)]\, d\eta. \end{aligned}$$

In these equations we designate the depth inference system by a two-dimensional layer A, the excitatory field by a two-dimensional layer M, and the inhibitory pool by a one-dimensional layer U. The spatial coordinates of layers A and M are retinal angle q and disparity d. The spatial coordinate of layer U is retinal angle q. The point potentials M(q,d,t) and U(q,t) in layers M and U depend on both time t and the spatial coordinates. Internal potential in M is converted to an external potential (or firing rate) by a saturation-threshold function f. Each point in layer M receives excitatory stimulation from the depth inference system A through the gain K_a, and from neighboring points through the spread function w_m. Similarly, the point potential in the inhibitory layer at time t is U(q,t), internal potential is converted to external potential by a simple threshold function g, and mutual excitation between points in U is described by the spread function w_u. Excitatory input to U at retinal position q comes through the gain K_u and is the integral along the disparity coordinate of all excitation at position q in the excitatory field. In turn, layer M receives an inhibitory signal from the inhibitory pool through the gain K_m. The rates of change of potential in layers M and U are governed by the time constants T_m and T_u.

Our model extends the Amari/Arbib scheme, denoted by eqs. (1) by employing two such processes, one receiving monocular depth cues from lens accommodation and the other receiving binocular depth cues from disparity matching. The two processes interact to reinforce similar depth estimates by means of cross-coupling pathways between their excitatory fields. The full depth perception model of Fig. 13 is represented analytically by the following four integro-differential equations:

$$\begin{aligned}
T_m \dot{M}(q,d,t) &= -M(q,d,t) + \iint w_m(q - \zeta, d - \eta)\, f[M(\zeta,\eta,t)]\, d\zeta\, d\eta \; + \\
&\qquad K_{sm}\, f[S(q,d,t)] - K_m\, g[U(q,t)] + K_a\, A(q,d,t), \\
T_u \dot{U}(q,t) &= -U(q,t) + \int w_u(q - \zeta)\, g[U(\zeta,t)]\, d\zeta + K_u \int f[M(q,\eta,t)]\, d\eta, \\
T_s \dot{S}(q,d,t) &= -S(q,d,t) + \iint w_s(q - \zeta, d - \eta)\, f[S(\zeta,\eta,t)]\, d\zeta\, d\eta \; + \\
&\qquad K_{ms}\, f[M(q,d,t)] - K_s\, g[V(q,t)] + K_d\, D(q,d,t), \\
T_v \dot{V}(q,t) &= -V(q,t) + \int w_v(q - \zeta)\, g[V(\zeta,t)]\, d\zeta + K_v \int f[S(q,\eta,t)]\, d\eta.
\end{aligned} \tag{2}$$

In these equations, A represents the accommodative depth inference system, D represents the disparity inference system, M the monocularly driven excitatory field, and S the stereoptically driven field. The corresponding inhibitory pools are represented by U and V. Excitatory cross-coupling between the monocularly and binocularly driven processes is represented by additional terms providing stimulation from each point in layer S to the spatially corresponding point of layer M through gain K_{sm}, and from M to S through gain K_{ms}.

Eqs. (2) provide a complete analytical description of the model. For a more complete explanation of the basis for this representation the reader is referred to Appendix A.

Methods

Computer simulation

Computer simulation was used to do the extensive testing and experimentation required for proper evaluation of the model. The equations representing the model were translated into an algorithm, coded in Pascal, and solved numerically on a VAX11/780. The simulation algorithm was fully implemented on the computer for both eyes but for only the left side of the brain. It is imbedded in a software system that provides an interactive graphics interface for constructing two-dimensional test scenes, tuning the simulation's parameters, and producing various displays of the simulation's state. Appendix C provides a detailed description of this system.

In the simulation system, simple optical equations are used to project a test scene onto two simulated semicircular one-dimensional retinas and to infer the initial accommodative and disparity depth cues. Depth cues from disparity are calculated directly by shifting the images on the two retinas with respect to each other and comparing corresponding points. Consistent with constraint 3), that vergence cues should not be employed, relative shifts of up to 50 percent of the binocular visual field are used. Accommodation cues are approximated using a Gaussian function centered about the correct depth of a point of stimulation, to generate a set of estimates that is broadly spread in the in depth (disparity) direction. Details of the optical equations are provided in Appendix B.

Eqs. (2) were simplified for the computer simulation. The simulation represents the two-dimensional layers M, S, A, and D as two-dimensional arrays and the inhibitory layers U and V as vectors. We made the assumption that the excitatory spread in the disparity direction of the two-dimensional layers would be small compared with both the spread in the retinal position direction and the coarseness of the discrete arrays. By this assumption, the double convolution integrals of eqs. (2) were reduced to single integrals, and the spread functions w_m and w_s were reduced to functions of only distance along the retinal position axis. Similarly, we assumed that the excitatory spread in the inhibitory pools would be small compared with both the spread in the excitatory layers and the coarseness of the discretization. By this assumption, the spatial convolution integrals were removed from the equations describing the inhibitory layers. After making these simplifications, eqs. (2) are reduced to

$$\begin{aligned}
&\tau_m \dot{M}(q,d,t) = -M(q,d,t) + \int w_m(q - \zeta)\, f[M(\zeta,d,t)]\, d\zeta + \\
&\quad K_{sm}\, f[S(q,d,t)] - K_m\, g[U(q,t)] + K_a\, A(q,d,t), \\
&\tau_u \dot{U}(q,t) = -U(q,t) + K_u \int f[M(q,\eta,t)]\, d\eta, \\
&\tau_s \dot{S}(q,d,t) = -S(q,d,t) + \int w_s(q - \zeta)\, f[S(\zeta,d,t)]\, d\zeta + \\
&\quad K_{ms}\, f[M(q,d,t)] - K_s\, g[V(q,t)] + K_d\, D(q,d,t), \\
&\tau_v \dot{V}(q,t) = -V(q,t) + K_v \int f[S(q,\eta,t)]\, d\eta.
\end{aligned} \tag{3}$$

Eqs. (3) provide a complete description of the continuous model as it was approximated by the simulation program. However, we wanted to test the model separately against both prey-like and barrier-like visual data in order to emulate the presence of these two channels in the visual system. Thus, the visual input to the simulation was broken up into separate "bug" and "barrier" channels, with the "bug" information passed to one complete version of the simulation and the "barrier" information passed to a second version. The two simulations were then run in parallel.

Visual input

A standard test scene was used for most of the experiments with the computer simulation. The configuration of this scene is shown in Fig. 16. The simulated animal is directly facing and about 20 cm. from the center of a paling fence that is oriented perpendicular to its body axis. Prey objects are located at two different positions behind the fence. This scene was chosen as a composite of typical configurations used in behavioral experiments [Collett, 1982; Collett & Udin, 1983].

When processed through the accommodation and disparity-matching depth-inference mechanisms, the standard test scene of Fig. 16 produces the input planes shown in Figs. 17a-d. These planes are depicted in the retinal-angle (q) vs. disparity (d) coordinate system of the model. Surface A shows the extent of the spread of the depth estimates provided by the simulation of accommodation. Surface D shows the ambiguity, due to spurious matching, of estimates provided by disparity. Figs. 6e-f show how the monocular estimates would appear if transformed back into the external Cartesian coordinate system and overlaid upon the original scene. They give only a coarse visual feel for the extent of uncertainty in the input but provide a means for comparison with the relatively fine discrimination achieved by the model in the studies presented below.

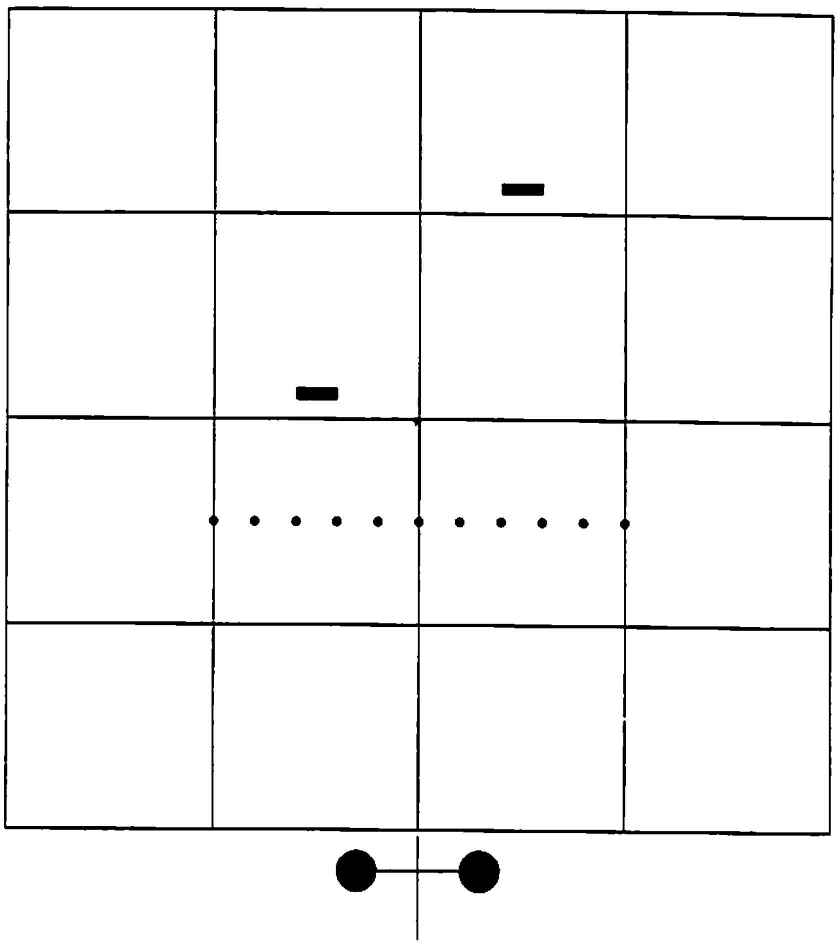
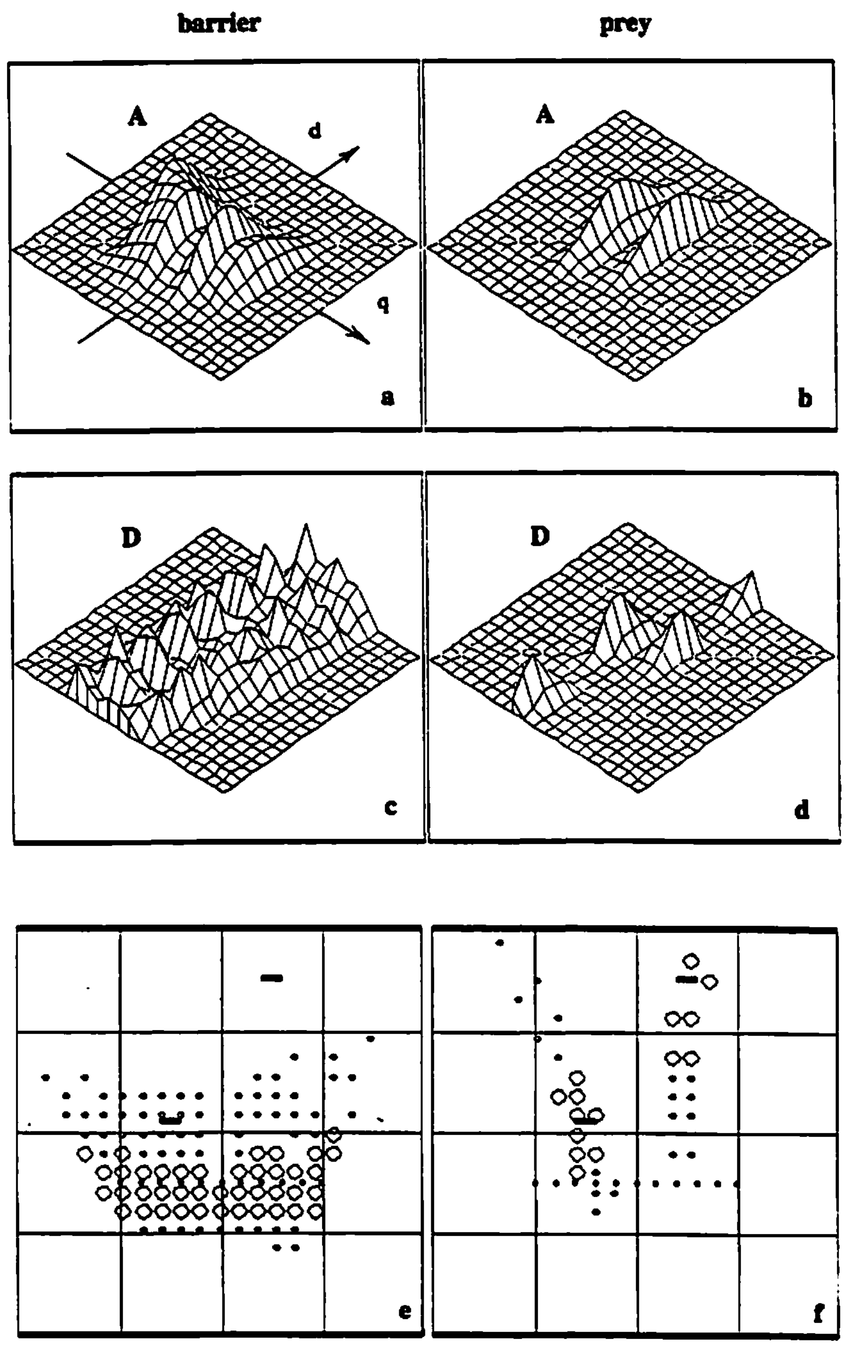


Fig. 16 - Standard Test Scene.

The visual input for most simulation runs was derived from the scene depicted here in top-view. Each grid element represents a 10 cm. by 10 cm. area. The simulated frog or toad is indicated by the icon below the grid area. The circles on the icon's cross-bar indicate eye position. Within the grid, the two rectangles represent prey and the dots represent the posts of a paling fence interposed between the animal and its prey. The simulation considers only the simplified case of one-dimensional retinas. Retinal images are formed by taking a horizontal slice through the scene.

Fig. 17 - Monocular and Binocular Inputs - Standard Scene.

These graphs show the initial monocular and binocular depth-estimates inferred from the standard test scene. The left-hand column of figures represents the barrier process and the right-hand column the prey process. Surface A represents the accommodative (monocular) depth inference system, and surface D represents the disparity (binocular) depth inference system. (a) through (d) are in the retinal position, q, vs. angular disparity, d, coordinate system. In these figures the height of the curve above the grid-like plane indicates the estimated likelihood that the particular disparity is the correct one for the corresponding retinal position. (e) and (f) show monocular estimates displayed in the original spatial coordinate system and superimposed upon the test scene. Relative estimate strength for a particular portion of the visual field is encoded by the size of the small "oval" at that position in the field.


barrier
prey
A
d
q
a
A
b
D
c
D
d
e
f

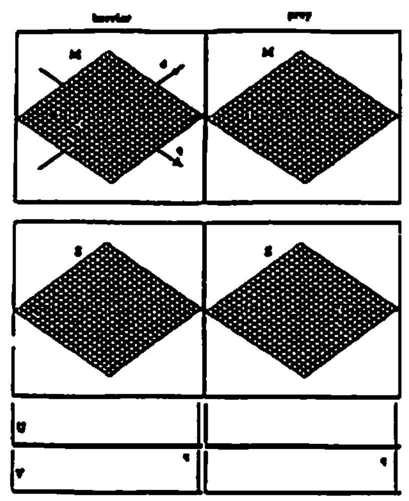
Results

Initial validation

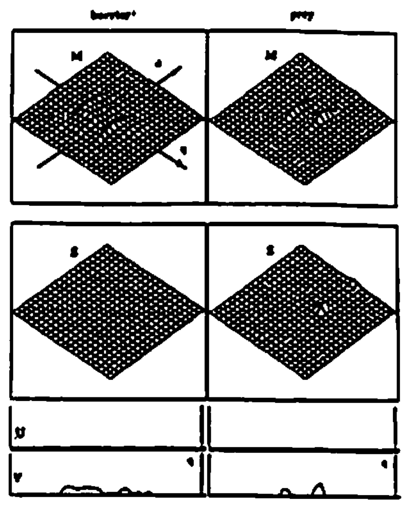
In order to demonstrate the simulation's convergence characteristics a time course from initial state to convergence is traced in Fig. 18. In this figure the monocular excitatory fields M and the stereoptic excitatory fields S are displayed as three-dimensional grids. Activity in the corresponding inhibitory pools U and V is displayed in the form of line graphs just below each column of two grids. The displays are temporally spaced at intervals of 2 time-constants of the excitatory field and thus represent a total of 10 time-constants of simulated activity. Although this time cannot be directly tied to actual processing time in neural tissue, the bench-mark of 10 time constants serves as a basis for comparison with the behavior of the model when tested, later, under purely monocular stimulation. The initial lack of excitation in the neural fields is depicted in Fig. 18a. The gradual build-up of excitation and the corresponding response in the inhibitory pools may be noted in Figs. 18b-c. In Fig. 18d the growth of inhibition has begun to shrink the widths of the excited areas of the fields. Figs. 18e-f show further steps towards reaching a satisfactory depth segmentation. The fields driven monocularly via accommodation cues and binocularly via disparity cues tend towards virtually identical states. Field excitation is maintained only in narrow bands along the disparity axis but shows significant spread in the retinal angle direction as nearby elements reinforce each other. At, all excited retinal positions the potential in the inhibitory pool reaches a level high enough to suppress exitation for all but one disparity.

Fig. 18 - Time Course of the Simulation - Standard Scene.

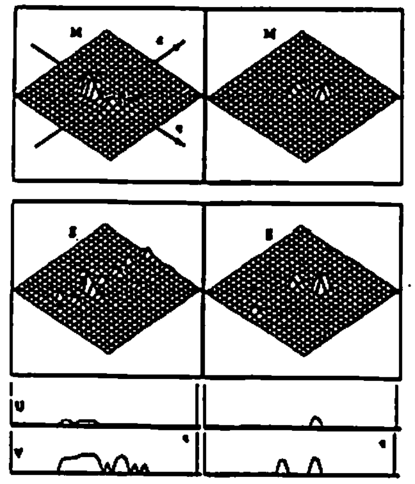
The time course of a simulation run is mapped from its initially inert state (a) to a satisfactory depth segmentation (f). Inputs were as shown in Fig. 17. Surfaces represent the level of excitation in the monocularly driven field M and the stereoptically driven field S. The retinal position, q, vs. disparity, d, coordinate system is indicated on the upper-left-hand grid in each figure. Line graphs of the level of activity in the inhibitory pools U (monocular) and V (binocular) as a function of retinal position, q, are given below each set of grids. Successive figures are temporally spaced 2 field time-constants apart. The elapsed simulation time represented is 10 time-constants.



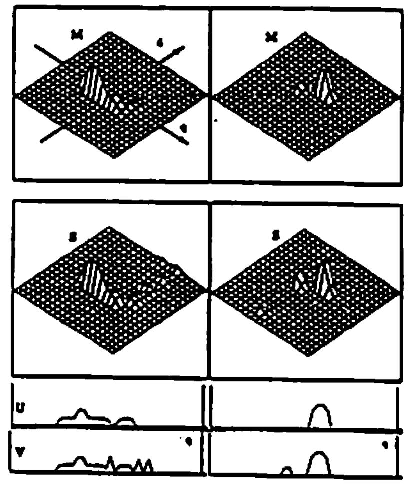
a



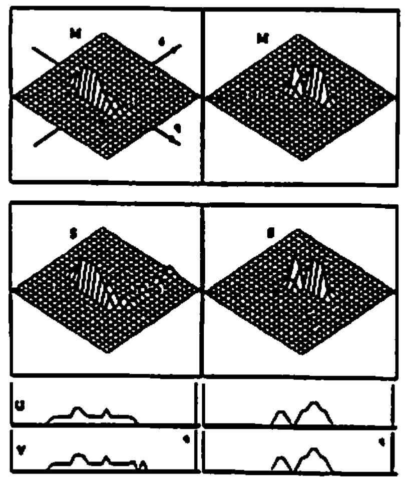
b



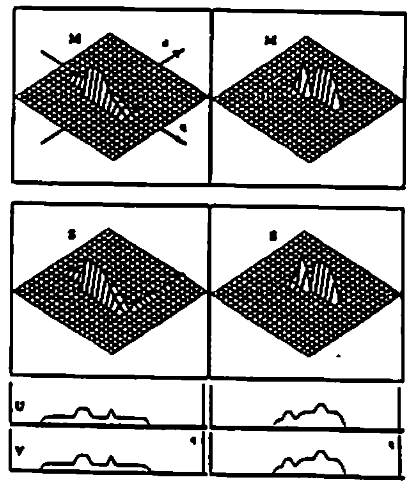
c



d



e



f

Fig. 19 shows the result of transforming the equilibrium state of the monocularly driven field back into Cartesian coordinates. This display is shown overlaid upon the test scene and indicates the extent to which the model was successful in constructing a depth map of its spatial distribution. The decrease of depth acuity with increasing distance may also be noted by comparing the estimates for the two prey.

Effect of lateral excitatory spread

Both the performance and the time-to-convergence of the simulation are strongly affected by the strength of the lateral connections in the excitatory fields. A series of runs was made in which the net gain of the spread function w[2] was varied from its nominal setting w_0. During these runs, performance characteristics and time to convergence were judged visually by watching a graphical display of the simulation. Fig. 20 plots observed convergence time, as a function of the net-strength of the excitatory spread, when the standard test scene was provided as input. Four regions where performance was judged to be qualitatively different are indicated at the bottom of this figure. In region I the excitatory gain was so low that convergence was slow, and no lateral spread of excitation beyond a given point of stimulation was noted. Excited points in the field remained below saturation level, with the level of excitation determined mainly by the strength of the input at a given point. In region II overall performance was judged to be best.

[2] Although eqs. (1) show two spread functions, w_s and w_m, we kept these functions identical in all of the reported simulations. Here we use the symbol w as a "shorthand" to denote both of these functions.

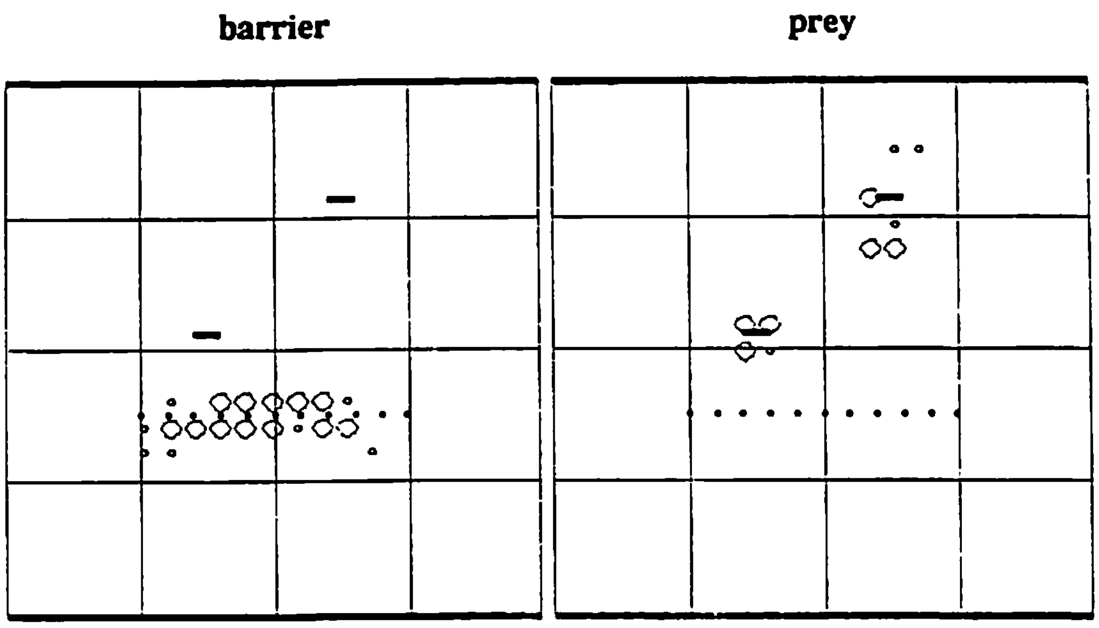
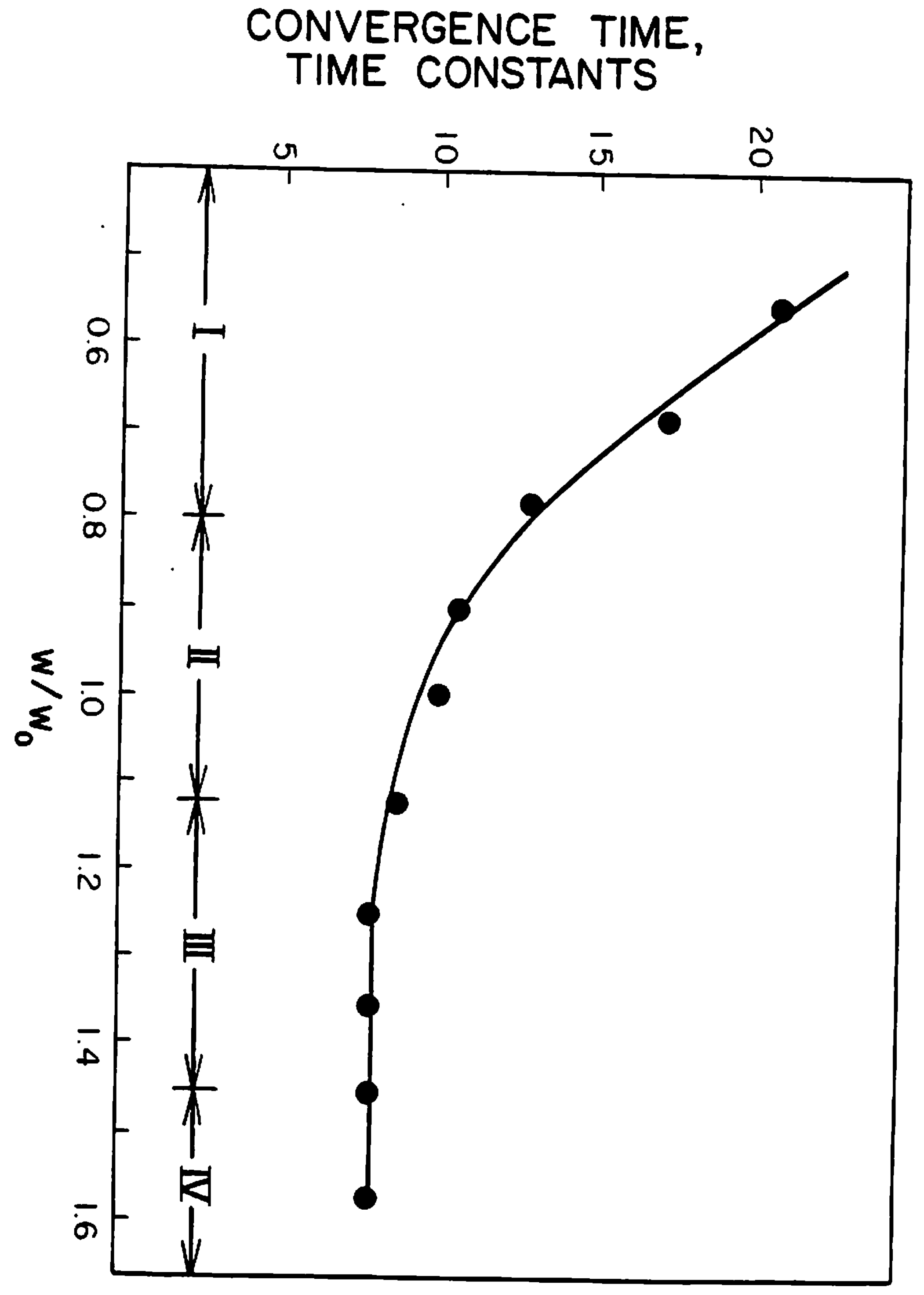


Fig. 19 - Scene Reconstruction from Depth Segmentation.

The final depth segmentation shown in Fig. 18f was used to produce the reconstruction shown here. Both the barrier and the two "bugs" were correctly spatially located. The reconstruction is shown over-laid upon the original scene for ease of comparison. The relatively broad spread, in the depth direction, of the estimate for the rear "bug" is due to the reduction of fine depth discrimination ability with distance.

Fig. 20 - Convergence Time vs. Strength of Excitatory Spread.

The model's speed-of-convergence and performance characteristics are highly dependent upon the net gain of the function describing spread of excitation in the excitatory fields. The horizontal axis is the net gain w of this spread-function, normalized to the nominal gain w_0 used for the other experiments described in this chapter. The vertical axis is time-to-convergence, measured in excitatory-field time-constants, when the standard test scene is used as input. Convergence and performance were judged visually by watching a graphical display of the model's state. The curve is divided into four regions to indicate gain settings for which performance was judged to be qualitatively different. In region I depth resolution was acute, and spontaneous spread of depth segments did not occur, but the level of excitation was highly dependent upon stimulus strength. In region II the model gave the best overall performance. Depth resolution was most acute, and convergence time was near optimal. However, spontaneous lateral spread of excitation did occur. Regions III and IV had the best convergence time but the uniqueness constraint was violated. In region III there was a tendency for points at nearby depths and the same retinal angle to simultaneously sustain excitation. In region IV points at quite different depths remained excited for a given retinal position.



CONVERGENCE TIME, TIME CONSTANTS
5
10
15
20
0.6
0.8
1.0
1.2
1.4
1.6
W/W_0
I
II
III
IV

Convergence speed was near optimal and excited points tended to be in saturation. However, in this region excitation was high enough to spread laterally to points not receiving external stimulation. Convergence was most rapid in regions III and IV, but here the uniqueness constraint was often seen to be violated. In region III neighboring points at the same retinal position often remained simultaneously excited, thus degrading depth-resolution. In region IV there was a more dramatic break-down of uniqueness, with multiple disparities at a single retinal position being above threshold.

Depth segmentation

Because of the relationship between model performance and excitatory spread, it is not possible to tune the model for rapid convergence without losing the ability to interpret its output as a point-wise reconstruction of the visual scene. With the model tuned for rapid convergence, the spread of excitation caused by the lateral connections in the cooperative fields results in a segmentation of the scene into depth-regions (Fig. 21). The breadth of each segment is dependent upon the length of the simulation run and the placement of objects in the scene. Segments grow out laterally on either side of a point of stimulation. Growth of a segment is constrained only by 1) the physical boundaries of the excitatory field or 2) intersection with a segment growing out from a different point of stimulation. If the model were tested against visually rich stimulation, such as a random-dot stereogram, the intersection of segments would be ubiquitous and segments would be prevented from growing much beyond the range of initial stimulation. The segmentation effect is seen only because of the sparsity constraint 5). Because of

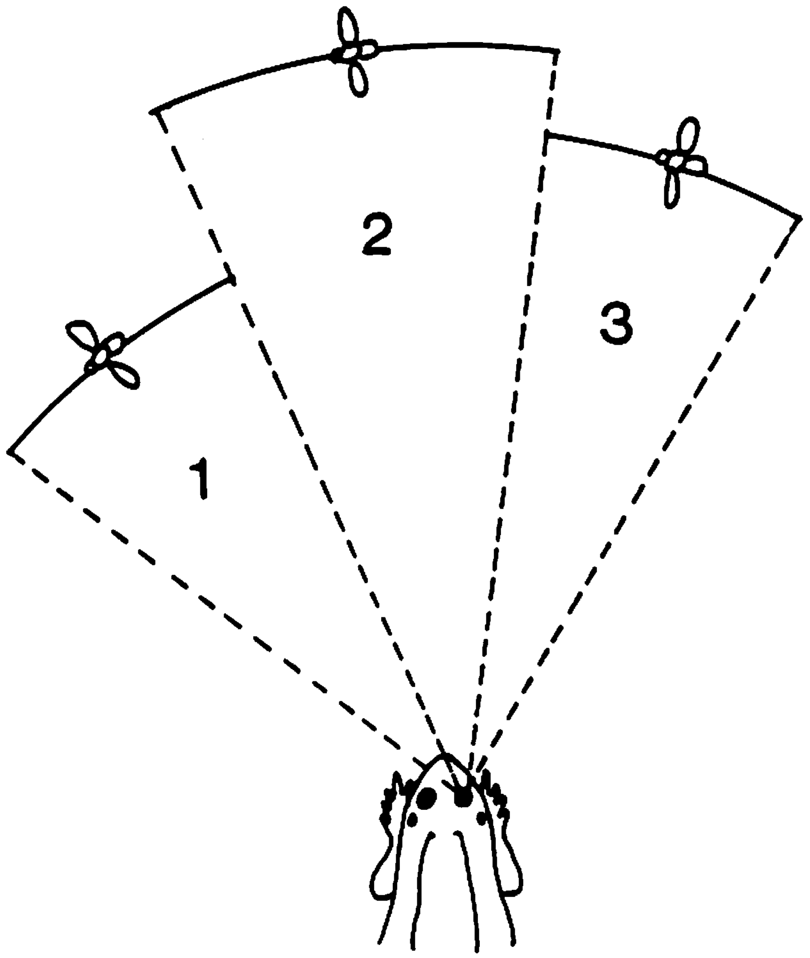


Fig. 21 - Visual Field Depth Segmentation Produced by the Model.

Instead of producing a depth-map of each point in the entire visual space, the model produces a map which is really a segmentation of this space into depth-regions. Each attention-fixating point in the visual field (such as a prey object) will result in the addition of a segment. Thus, with the three bugs shown in front of the animal, the scene is divided into the three numbered depth regions. It is hypothesized that for prey-catching the animal will determine the correct visual angle from the current retinal stimulus, and will then infer depth by reference to the corresponding portion of the depth-map.

the segmentation effect, the spatial coordinates of a visual object cannot be determined by means of the model's output alone. However, spatial location can be determined by 1) the extraction of visual angle from direct retinal stimulation, and 2) sampling the depth map at this angle to determine depth. The segmentation effect has important behavioral implications to be discussed later.

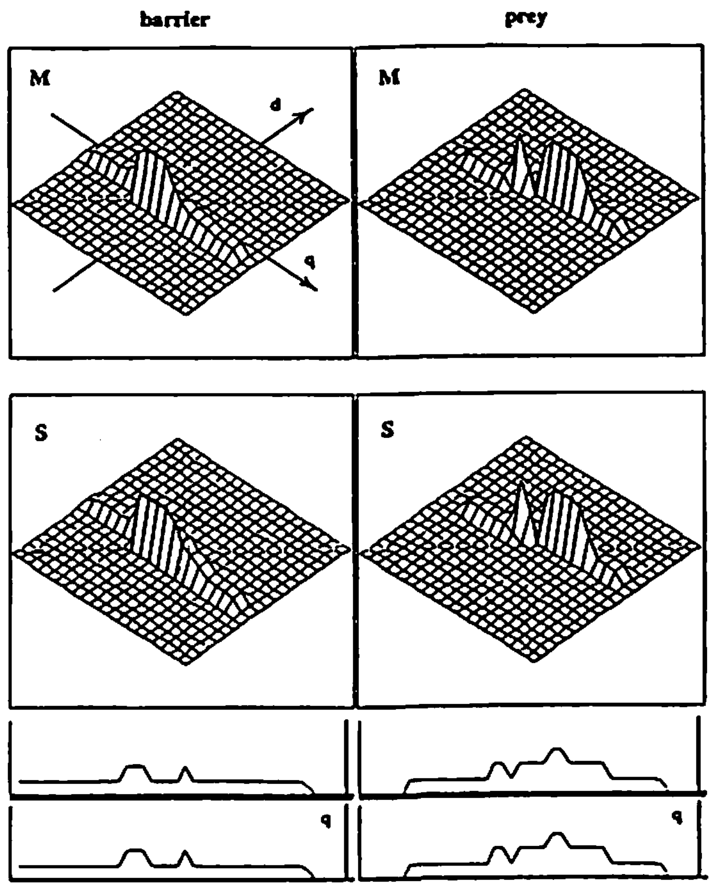
Fig. 22 provides an illustration of the depth segmentation effect, and how the resulting problem of poor angular resolution can be overcome. In order to dramatize the segmentation effect the simulation was was allowed to run twice as long as the simulation of Fig. 18 (20 field time-constants). The final state of this run is shown in Fig. 22a, and its transformation back into external coordinates is shown in Fig. 22b. Finally, Fig. 22c shows that, even though the depth segments are greatly exagerated, objects can still be successfully localized by sampling the depth map at only those retinal positions currently receiving stimulation.

Effects of lenses and prisms

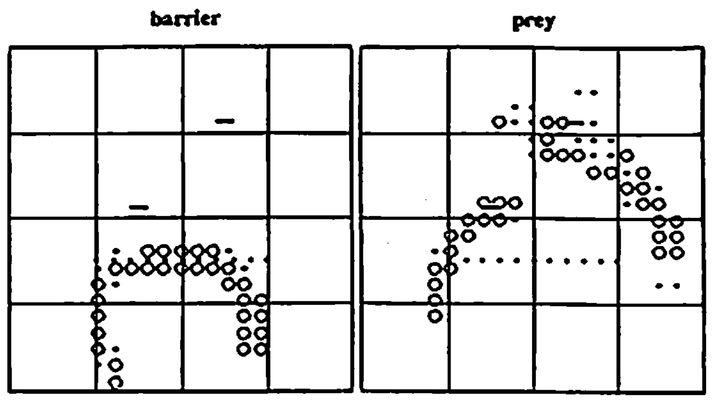
Fig. 23 shows the depth maps produced by simulation runs employing the same input scene but using simulated interposed lenses (Fig. 23a) and prisms (Fig. 23b). The simulated lens and prism strengths were chosen to produce a 20 percent shift in the disparity coordinate of the corresponding monocular (lenses) or binocular (prisms) input plane. The results for prey stimuli show that the shift in the depth segmentation produced by lenses is quite small compared with that produced by prisms. We now describe how parameters were chosen to affect this bias.

Fig. 22 - Object Localization from Depth Segmentation.

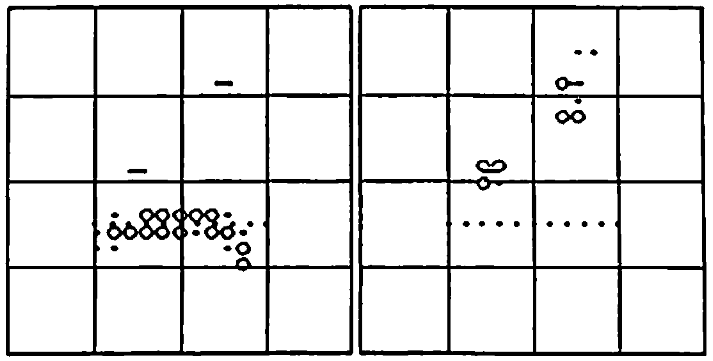
By doubling the simulation run-time from that represented in Figs. 18 and 19 the spread of field-excitation was increased in the retinal angle direction (a). The resulting segmentation effect is apparent in the reconstruction shown in (b). Nevertheless, objects can still be accurately localized by sampling the depth-map only at retinal positions currently receiving direct stimulation (c).



a



b



c

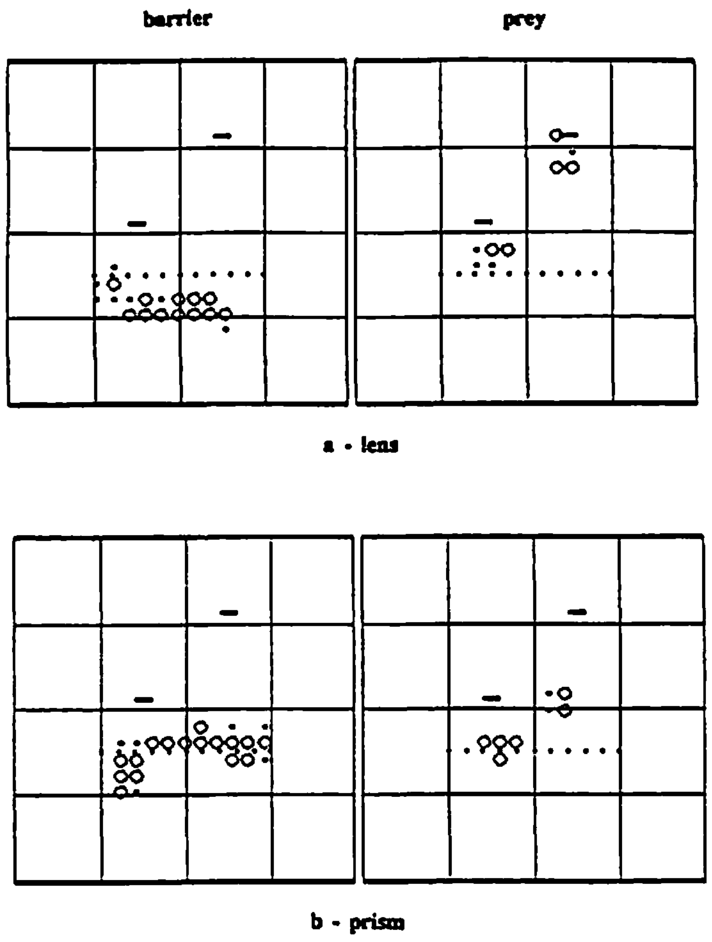


Fig. 23 - Lens and Prism Effects.

Simulated lenses, set to produce a 20% shift in the disparity coordinate of the monocular input, were used to obtain the reconstruction shown in (a). Only a small shift in the model's depth estimate for prey stimuli occurred. The barrier estimate was more grossly affected. Simulated prisms, producing a 20% disparity shift in the binocular input, were used to produce (b). The large shift apparent in the prey depth-estimate compared to that obtained with lenses is consistent with behavioral data. The different lens and prism results obtained for barriers are explained by the close spacing and spatial periodicity of the fence-posts. Currently, no behavioral data exist for this type of stimulus.

The biasing of the model to favor binocular cues in determining depth-of-prey is very sensitive to the relative strengths of the binocularly and monocularly derived inputs. Experimentation showed that no internal model parameters could affect this biasing more than the relative weighting of the input gains K_a and K_d. In order to show this result more explicitly, a series of experiments was run with an input scene consisting of a single prey placed in the center of the scene. With lens strength held constant at 20 percent, the ratio of binocular input strength K_d to monocular input strength K_a was varied over two log units (from 0.1 to 10.0) in such a manner as to hold the net input strength ($K_d + K_a$) constant. Error in prey depth estimate as a function of this ratio is plotted as a dashed line in Fig. 24. Experiments utilizing 20 percent prisms, instead of lenses, produced the results plotted as a solid line in Fig. 24. Setting the ratio K_d/K_a to about 3.0 will reproduce the results obtained by Collett [1977], showing a major binocular effect and only a minor monocular effect. The steep slope of the prism curve compared with the more shallow slope of the lens curve can be accounted for by the difference in acuity of binocular versus monocular cues.

Barrier depth resolution

Depth resolution of prey in the presence of lenses and prisms is consistent with experimental results. There have been no experiments on the resolution of barrier depth, so the results of our simulations may be seen to yield interesting predictions. In the simulations depicted in Fig. 23 the lenses produced a large shift in the depth segmentation whereas the shift due to prisms was smaller and in the opposite direction. This effect is further explored in the experiments depicted in

Fig. 24 - Depth Error vs. K_d/K_a.

The dashed line shows the error, in centimeters, of the calculated position of a prey, versus the ratio of the strength of disparity cues (K_d) to accommodation cues (K_a), when viewed through 20% lenses. The solid line shows a similar curve obtained using 20% prisms. Data for the curves came from simulations using, as visual input, a scene consisting of a single prey located 22 cm. from the simulated animal along its midline.

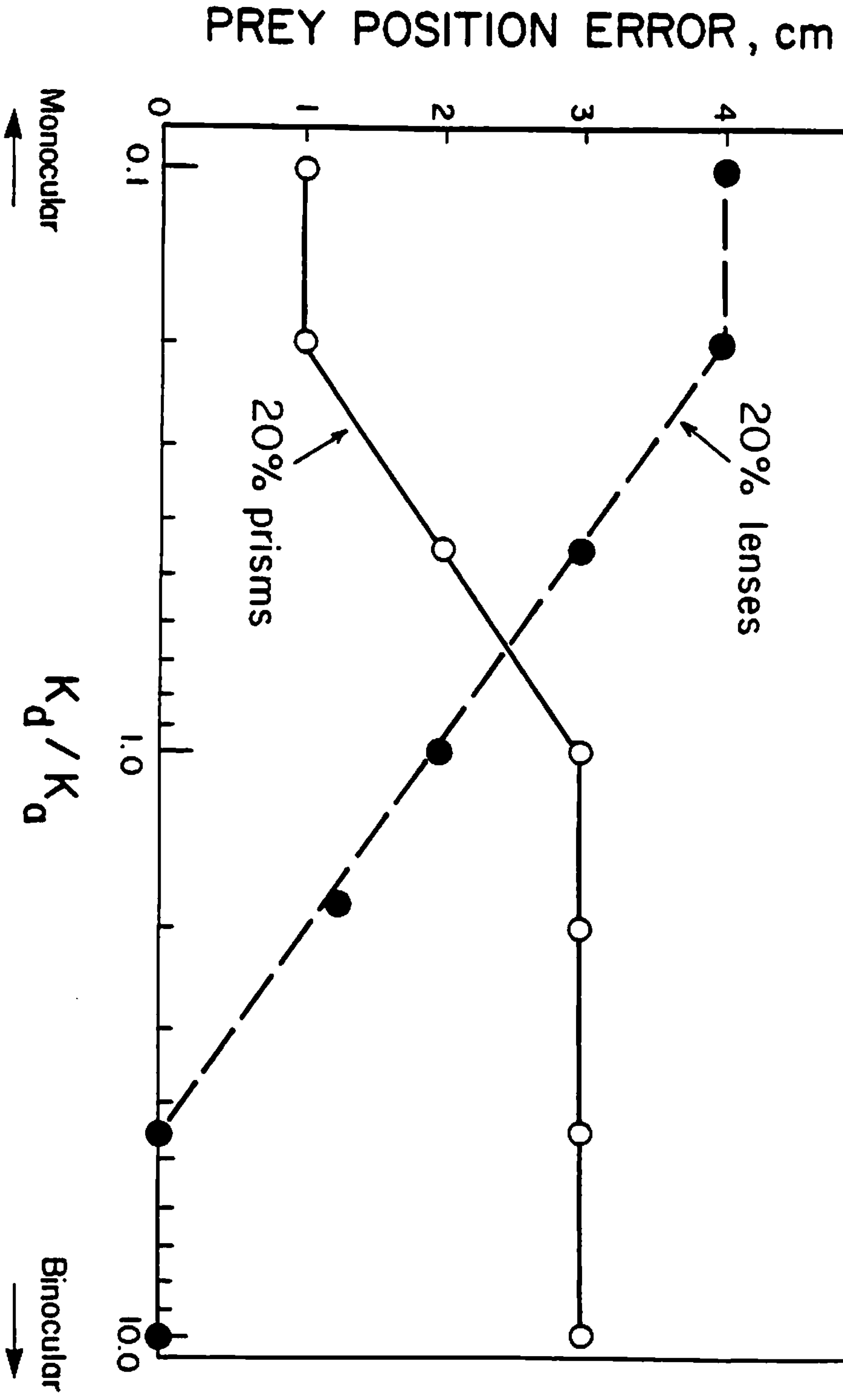

PREY POSITION ERROR, cm
0
1
2
3
4
5
0.1
1.0
10.0
20% lenses
20% prisms
K_d / K_a
Monocular
Binocular

Fig. 25. This figure shows the depth-analysis for a single centrally-placed fence and a range of lens and prism settings. We see that the effect of lenses is to shift the depth estimate closer to the animal as the lens strength increases. However, the effect of increasing prism strength is less consistent. Depth estimates shift forward and back, often fragmenting into several depth regions. This effect can be explained by the close disparity spacing of the spurious binocular depth estimates derived from the repetitive fenceposts. Any mismatch between monocular and binocular estimates that exceeds this disparity spacing leads to a reinforcement of one of the spurious estimates rather than to a reinforcement of the correct one. For more visually sparse data, such as prey stimuli, reinforcement of spurious estimates is unlikely.

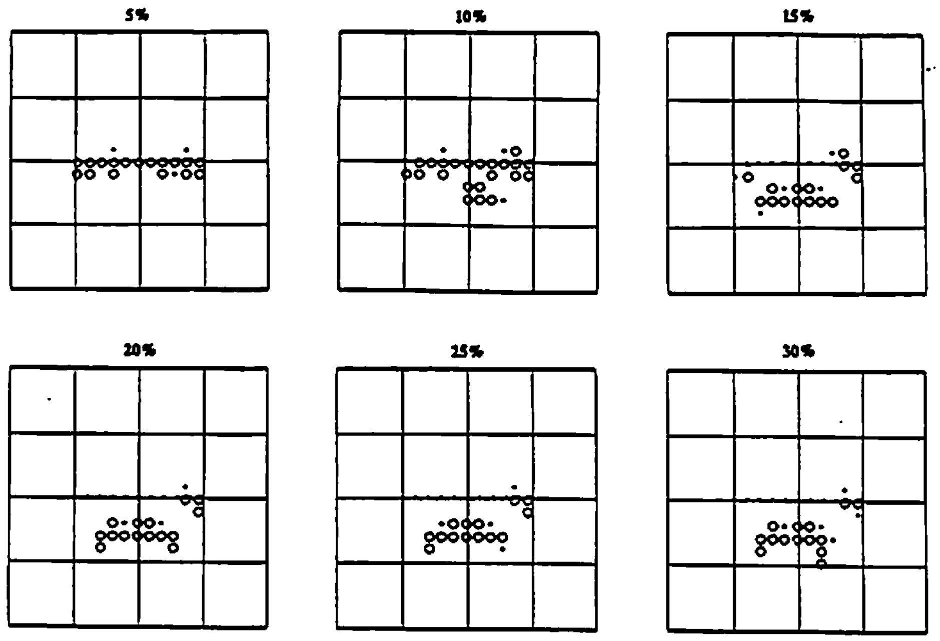
Another experiment was done to examine the response of the model when confronted with two fences. Fig. 26 shows the model's response to a scene containing two fences placed one in front of the other. The simulation resolves portions of each fence but does not completely resolve both. This is because, in conformance with the uniqueness constraint 1), the model was tuned so that excitation cannot be sustained at multiple depths for a single retinal position.

Monocular response

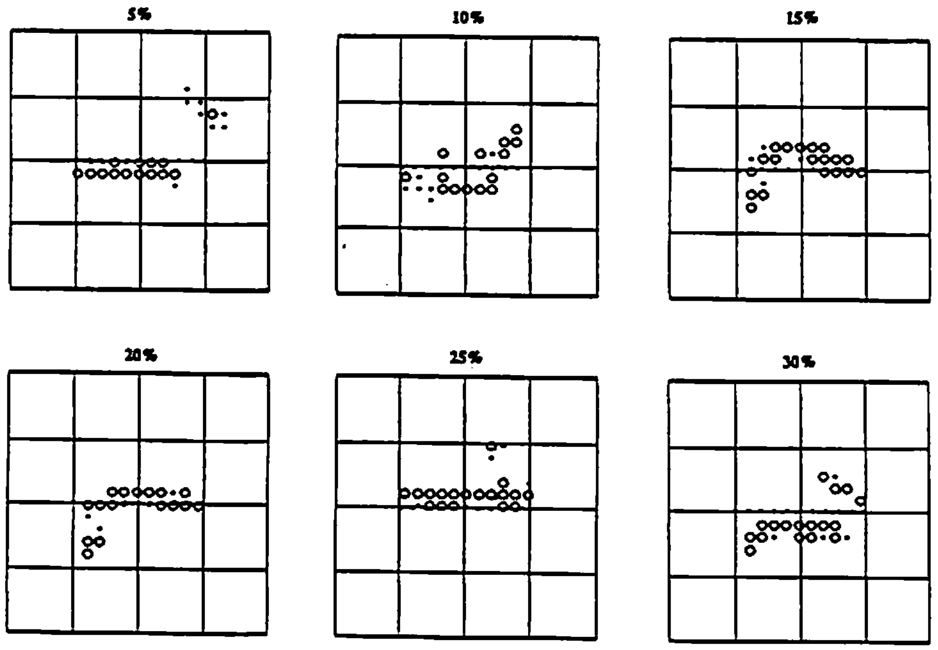
A final experiment was done to test the model's performance when deprived of binocular depth cues, thus replicating a monocular animal. Fig. 27 shows the results of this experiment. The figure shows that with the binocular input strength K_d set to zero the model can still produce a correct depth segmentation. Although accuracy was not noticeably impaired in this simulation run (compare with Fig. 19),

Fig. 25 - Effects of Lenses and Prisms on Fence Depth Estimation.

Estimates of the depth of a fence for various lens (a) and prism (b) strengths. As lens strength increases the effect is simply to move the perceived depth plane of the fence closer to the animal. Increasing prism strength has a more variable effect. For some prism settings the fence is seen as being broken-up into several depth regions.



a - lens



b - prism

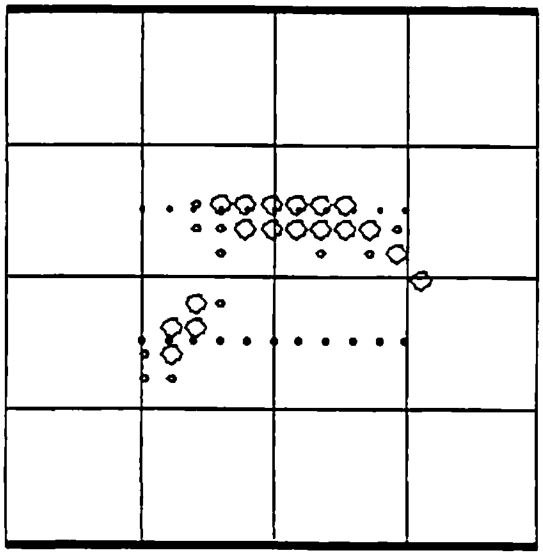


Fig. 26 - Two Fences.

Because of the uniqueness constraint it is not possible to completely represent two fences in two distinct depth planes. This figure shows the fragmentation of the depth segmentation which results from the simultaneous presentation of two superimposed fences.

---

the length of time required to reach a satisfactory segmentation was doubled, from 10 to 20 field time-constants. The slow-down in convergence time is due to the net reduction of input gain incurred by removing one of the model's input sources.

Discussion

One of our initial goals was to design a stereoscopic depth-perception model that conforms to the methodology used in previous cooperative models, but that employs accommodation cues, instead of cues from vergence, to resolve binocular

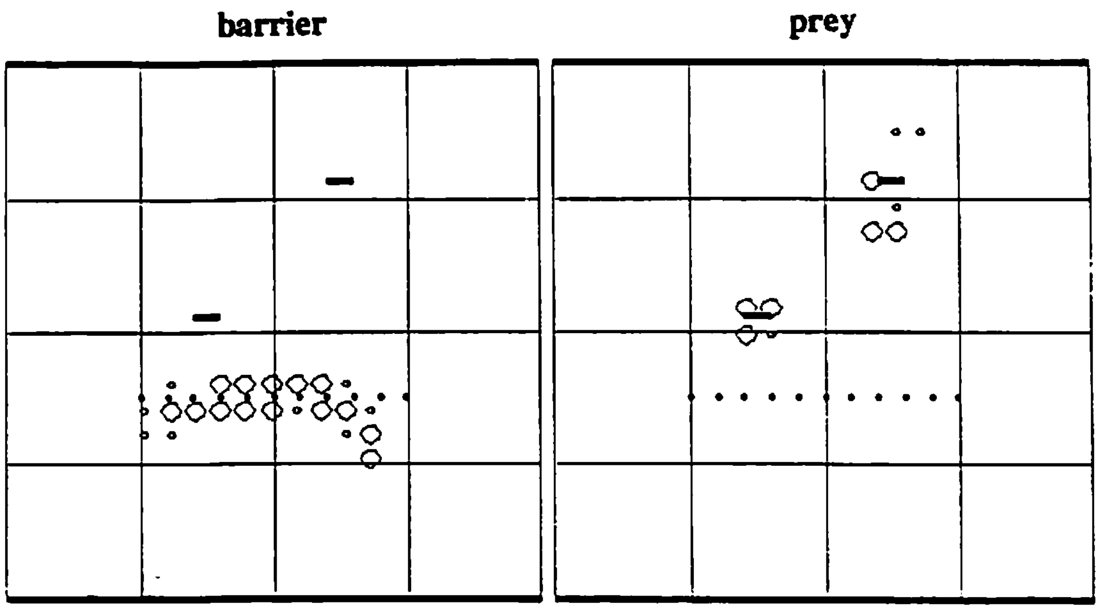


Fig. 27 - Monocular Response.

The response of the model when only monocular input is available was obtained by setting the binocular input gain K_d to 0.0. The resulting depth map compares well with that obtained from the full monocular/binocular model. However, the amount of time (20 field time-constants) required to achieve a satisfactory segmentation was double that of the full model.

ambiguity. A second goal was to define a model whose resulting depth maps are dominated by binocular depth cues, but that can still function when only monocular cues are available. Simulation runs have shown that the model is successful in meeting both of these goals. Since disparity matching is not confined to a small area around a point-of-stimulation, vergence cues do not play a role in the depth mapping process. Instead, the range of disparities to be considered is effectively constrained by accommodation cues. Disparity cues that match strong accommodation cues are much more likely to be selected than those that do not.

Also, the simulation can be tuned to provide an adequate depth mapping both with and without binocular information, and under this same tuning its depth mapping process is dominated by binocular cues when they are present.

The model differs from previous models in its use of accommodation to obtain depth cues. Earlier models were tested with only static stereoscopic images as input. Since our model uses the process of accommodation to obtain secondary depth cues it must be tested against a model of the input scene that includes the depth dimension. In this sense, it is more suitable for representing the depth resolution process of animals. For this same reason, its method of using accommodation cues to disambiguate binocular cues might be useful in a robotic depth perception system. Robots interact, not with static images, but with true three-dimensional scenes, and thus could benefit from the use of depth cues obtained from accommodation.

Inspiration for the model and tuning data for the simulations were taken from the literature on the visual system of frogs and toads. In particular, tuning of the relative weights of accommodation and disparity cues was guided by consideration of data on prey catching in toads [Collett, 1977]. We confirmed that the simulation can be tuned so that its discrimination of the spatial locations of prey-like stimuli is only slightly affected by the presence of moderate systematic error in the accommodative mechanism (lens effect), while remaining very sensitive to systematic binocular error (prism effect). Since the ratio of accommodation to disparity cues used to produce these results is only one of many possible choices, the model can be thought of as representing a spectrum of depth perception mechanisms that range from one totally dependent on accommodation cues to one that uses only disparity cues.

The model also leads to new predictions concerning the errors that frogs and toads might make in determining the depth of fence-like barriers. When both the monocular and the binocular mechanisms are perfectly in tune with each other, fence depth is correctly ascertained. However, a small detuning of either mechanism will lead to a breakdown in consistent discrimination capability. Further, there is a qualitative difference between the effect of lenses and the effect of prisms on this type of depth resolution. Lenses have the expected effect in shifting the depth estimate. However, prisms have a less predictable effect and tend to produce an image fragmented into several depth regions.

The use of accommodation to disambiguate, and not merely supplement, binocular estimates resolves a problem posed by Collett and Harkness [1982]. Their calculations show that in small animals the use of a weighted sum of binocular and monocular depth cues, rather than purely binocular cues, can produce only a small improvement in accuracy. For instance, in toads, the use of such a weighting scheme can yield only a 5 percent improvement. It is not clear, based on this analysis, what advantage frogs and toads gain by the simultaneous use of both types of cue. However, our analysis shows that if accommodation cues are used to help select among competing binocular cues, and not merely to refine accuracy, depth perception can be done with greatly enhanced speed and reliability.

In developing the model we assumed that frogs and toads are able to simultaneously maintain separate depth segmentations of "bug" and "barrier" features within the visual field. We also assumed that direct optic-nerve input provides the angular position of an object, while a depth segmentation map provides its depth. Arbib and House [1983] further exploit these assumptions in the development of models of the orientation and barrier negotiation behavior of frogs and toads.

CHAPTER IV

LOCALIZATION OF PREY

Summary

In this chapter we present an action-oriented model of the spatial localization of prey by frogs and toads. Instead of building a global depth-map we propose that the goal of catching a prey can lead a frog or toad to select a particular region of its visual world for special scrutiny. We suggest that the first step of the prey-catching sequence is not an overt movement. Rather, it is a covert movement to adjust the accommodative state of the lenses and thus lock the visual apparatus on to a stimulus.

We demonstrate how prey localization can be acheived rapidly and accurately by coupling prey-selection and lens-accommodation processes within a feedback loop. Information derived from prey selection supplies a setpoint for accommodation. In turn, adjustment of the lens modifies the visual input and can alter the prey selection process. The natural feedback of this goal-seeking system automatically corrects for the problem of ambiguity in binocular matching.

Although it is of general interest as a depth algorithm, we tie the model to the known anatomy, physiology and behavior of frogs and toads. We identify brain regions that could provide the neural substrates necessary to support the model's various functional stages and present experiments, with a computer simulation, that compare its functioning with animal behavior.

Introduction

Background

To date there is little evidence that frogs and toads have the anatomical and neuro-physiological features necessary to support a global depth-mapping process based solely upon binocular disparity. Unlike primates, they have neither the ability to verge their eyes nor a distinct foveal region of the retina. Thus, there is no means by which a portion of the scene can be fixated and analyzed via sharply-tuned neural disparity detectors. Instead, the eyes of frogs and toads are fixed [Grüsser & Grüsser-Cornehls, 1983], and receptors are comparatively evenly distributed over the retinal surface [Fite & Scalia, 1976]. The binocularly sensitive neuronal units that have been located in frogs and toads [Brown & Marks, 1977; Raybourn 1975] do not provide an output that is selective for disparity. Instead, the firing rate of a typical unit appears to be controlled by a summation of the input over the unit's receptive field.

Notwithstanding this lack of binocular machinery, it has been conclusively shown that toads prefer binocular over monocular depth cues when localizing prey in the frontal binocular visual field. Collett [1977], in a series of experiments using lenses and prisms to dissociate monocular and binocular depth cues, showed that when binocular information is available it dominates monocular information in the toad *Bufo marinus*. He was able to closely approximate observed prey-snapping errors with a formula that assigned a 94% weighting to binocular cues and only a 6% weighting to monocular cues.

Although binocular depth cues are preferred when available, both frogs and toads are able to utilize monocular lens accommodation cues to snap accurately at prey when deprived of binocularity. Monocular frogs [Ingle, 1976] and toads [Collett, 1977] show little reduction in accuracy when snapping at prey ipsilateral to their good eye. Additionally, experiments in which drugs were used to disrupt the normal action of the lens accommodation muscles have demonstrated that lens accommodation is the most likely source of monocular depth cues [Jordan et al., 1980].

In the previous chapter we investigated a mechanism of cooperativity between binocular and monocular cues to determine depth. The model developed in that study was successful in integrating these cues into a coherent scheme, but it utilized disparity detectors and its output was a global depth-map. The model presented in this chapter maintains the notion of the coupling of binocular and monocular mechanisms. However, it does not depend upon disparity detectors and, rather than computing a global depth-map, it localizes only a single point in space.

The prey-localization model is an outgrowth of a recent model developed by Collett and Udin [1983] to explain results obtained from lesion experiments. They noted that toads (*Bufo marinus*) with lesions to the isthmic nuclei maintained the ability to snap accurately at prey. Experiments utilizing prisms showed that, like intact animals, lesioned toads continue to use binocular depth-cues.

The finding that toads lacking their major source of tectal binocular information [Grobstein & Comer, 1983] still use binocular depth perception challenged the notion that a depth map could be computed in tectum. In place of local disparity matching, Collett and Udin suggested a process similar to that observed in mantids [Rossel, 1980]. In their scheme, each tectal lobe selects a particular visual object and passes its retinal position to a common brain region. This region then uses these two retinal angles to determine the spatial location of the object. Fig. 28 depicts this scheme and contrasts it with one based upon local disparity matching.

Although toads with lesions to the isthmic nuclei are as accurate as controls in striking at a single prey, they are subject to errors in binocular correspondence not seen in the controls. This result indicates that the role of the isthmic nuclei in this process is to provide cross-tectal relays that assist the two tectal hemispheres in agreeing upon the same visual object. If the isthmic connections are lost, then each tectal hemisphere will independently arrive at its own choice of stimulus and binocular mismatch will be probable.

An apparent weakness of Collett and Udin's model is that when presented with two prey stimuli, it chooses the prey nearer to the horopter. Their computer simulations showed that if the horopter surface, in relation to which the ipsilateral and contralateral visual fields are in register in tectum, is at 50 cm. (for a 3 cm. eye separation) then the chance of binocular mismatch is minimized. However, since the snap-zone of toads is well within 50 cm. [Ingle, 1970], placing the horopter at this distant location results in the selection of the more distant of two prey. Since actual animals prefer the nearer of two equally attractive prey, this result is inconsistent with behavior. Collett and Udin addressed this problem by suggesting that lens accommodation could play a role in bringing the nearer prey into sharper focus, thus making it more salient.

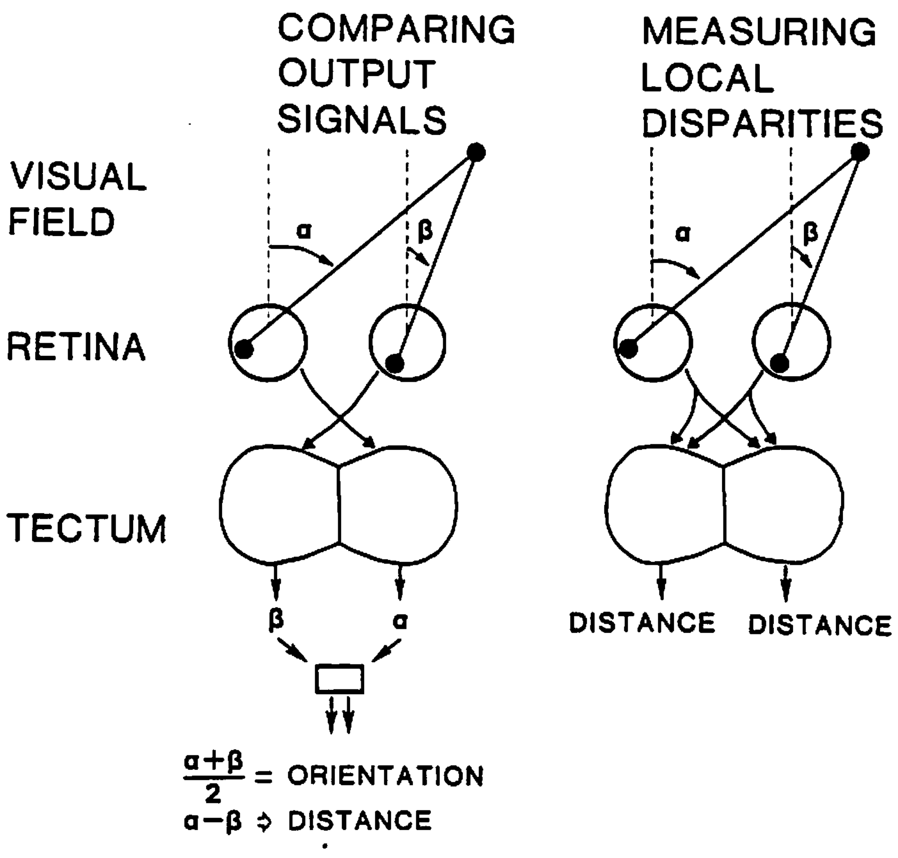


Fig. 28 - Depth from Selection vs. Depth from Local Disparity.

Two means of determining depth from binocular images are presented. On the right-hand side both tectal lobes build a depth map by measuring local disparities between points on the two images. Tectal output would then include an encoding of this depth. On the left-hand side each tectal lobe selects a particular point-in-space. Tectal output simply encodes these positions. If the two tectal sides choose the same point, the depth and orientation of this point can be readily determined from its positions on the two retinas.

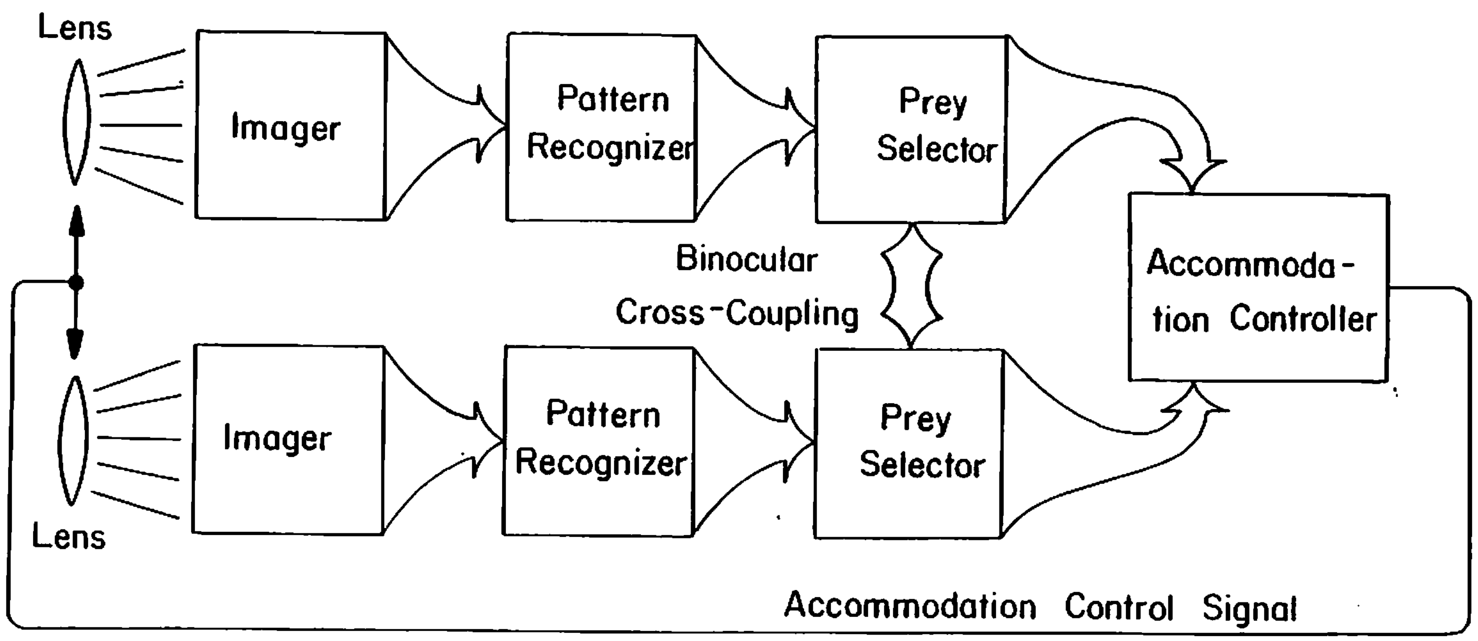
In the next section we modify and extend the ideas of Collett and Udin by proposing a model in which binocular prey-selection and lens-accommodation processes are tightly coupled in a feedback loop. The coupled system is shown to rapidly converge upon a single prey and, under a reasonable set of assumptions, will select the nearer of two prey.

Model overview

The functional components of the model consist of a binocular imaging system, pattern recognizers, prey selectors, and an accommodation controller. Fig. 29 depicts these components and how they are interconnected. The imagers produce an output pattern whose intensity at any position is dependent upon the crispness-of-focus of the image at that position. The pattern recognizers take their input from the imagers and, in turn, provide an output modulated by the degree to which each region of the image matches recognition requirements. It is assumed that the pattern recognition process is optimized when the image is in sharp focus. The outputs of the pattern recognizers project to the prey selectors. The selectors are responsible for identifying the region of the original image that corresponds to the strongest signal from the pattern recognizers. The two prey selectors are also binocularly cross-coupled to assist them in agreeing upon the same visual object. The outputs of the prey selectors are the image coordinates of the selected object. These coordinates project to the accommodation controller where they are used to compute depth from binocular disparity. This depth, in turn, is used to adjust the accommodative state of the lens.

Fig. 29 - Functional Diagram of the Prey Localization Model.

The functional components of the model and their interconnections are shown. Lenses are coupled so that they are accommodated to the same depth. Imagers produce visuotopically mapped output signals whose intensity is proportional to the crispness-of-focus at each image point. Pattern recognizers produce visuotopically mapped output signals that are strongest wherever the image most closely corresponds to the recognizer's matching criteria. Prey selectors take their input from the recognizers and from cross-coupled connections with each other. Their outputs are highly selective, giving a high weight to points receiving the maximum input stimulation and suppressing all others. Since the prey selectors take their input both from the pattern recognizers and from cross-coupling they are biased in favor of points receiving strong stimulation on both sides of the binocular system. The accommodation controller converts the weighted image coordinates from both prey selectors into an estimate of depth. It then uses this calculated depth to adjust the lenses.


Lens
Imager
Pattern Recognizer
Prey Selector
Binocular Cross-Coupling
Accommoda-tion Controller
Lens
Imager
Pattern Recognizer
Prey Selector
Accommodation Control Signal

Adjusting lens focus completes a feedback loop that, along with the binocular cross-coupling of the prey selectors, assists in the resolution of binocular ambiguity. If both prey selectors have chosen the same object in real space then the resulting depth estimate will be correct and the effect will be to bring this object into sharper focus. The improvement of focus will enhance the pattern recognition process and improve both the confidence and angular resolution of the recognizer's output. However, if the prey selectors have chosen different objects the depth estimate will be incorrect and the lens will be driven out of focus. This will affect activity in the pattern recognizers and their new outputs will alter the selection process.

If only a single object is present in the visual field, there will be no source of binocular ambiguity and the system will rapidly localize this object. If multiple objects are present, but at different depths, the action will be to bring one of these objects into clear focus, and again a correct depth estimation will be assured. The more difficult problem of multiple objects at the same depth will be resolved by the cross-coupling of the prey selectors. Thus, the only situations that will be ambiguous will be those in which the visual angle between two objects, at the same depth, is smaller than the resolution of the binocular cross-connections.

Methods

Computer simulation

A computer simulation of the model was developed so that experiments could be done to evaluate its performance. This simulation is implemented in Pascal on a VAX 11/780. Model results are displayed using a 512 × 512 pixel Grinnell color display. A menu-driven interactive graphics system, using this same display, allows the experimenter to adjust model parameters and test-scene configurations. Hard-copy images mimicing the interactive display are produced on a Symbolics laser-graphics printer by a batch version of the modeling system. Further details of the simulation system are given in Appendix C.

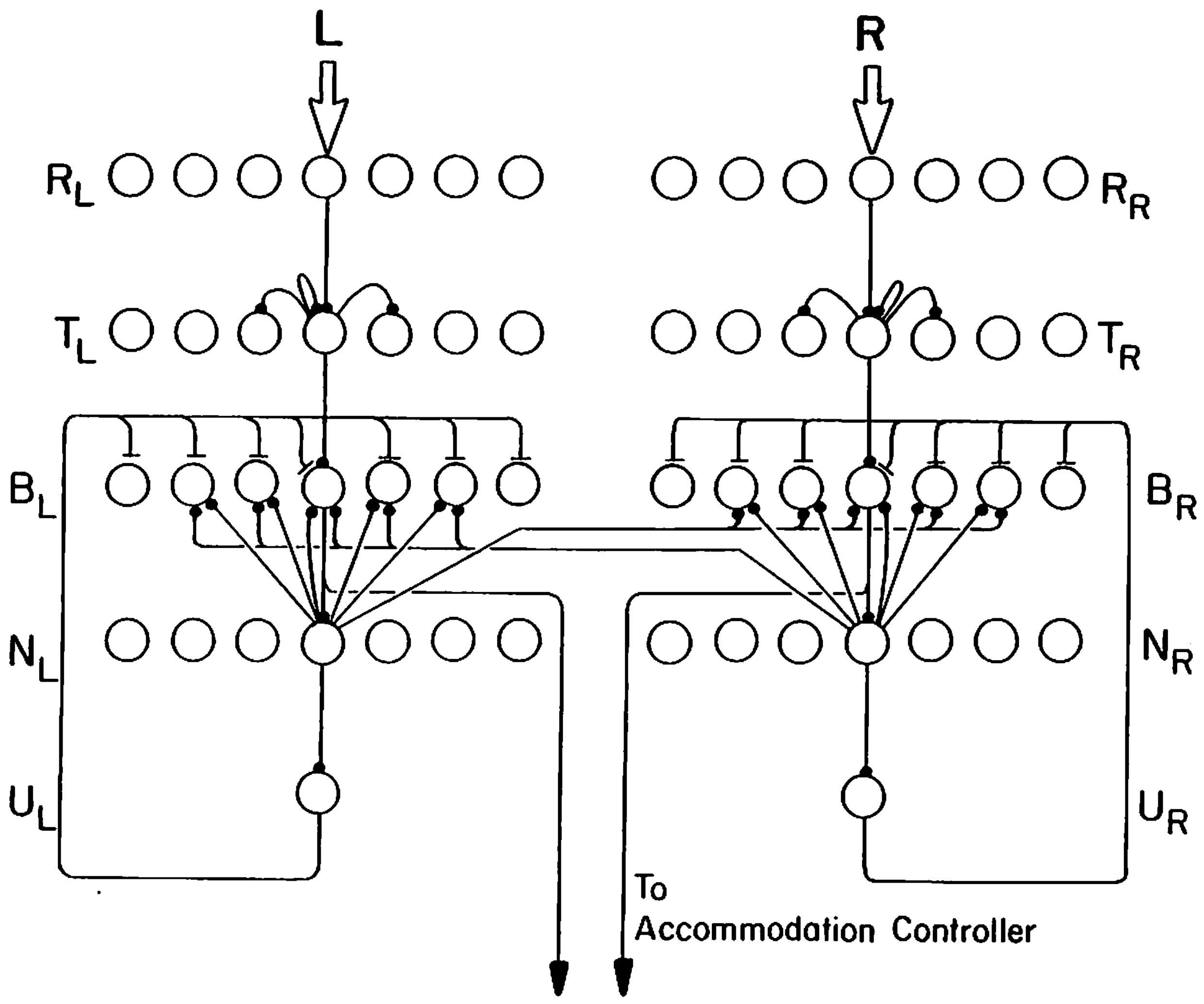
Several simplifications and assumptions were made when constructing the simulation and experiments. The test scenes used in all of our experiments consisted of either one or two elongated textureless stimuli in the frontal visual field. We did not attempt to represent a background. The simulation treats only two-dimensional scenes projected onto one-dimensional image planes. We assumed that the lenses are coupled so that both lenses are always accommodated to the same depth.

Mathematical description for the simulation

The functional elements of the model are represented as layered networks of cellular components. This constraint was adopted so that computational algorithms would be suggestive of possible underlying neural mechanisms. Fig. 30 shows these layers and their interconnections. For simplicity, only output connections from the central cell of each layer are shown. Similar connections are made for all other cells in the corresponding layers. The mathematical description of the model

Fig. 30 - Implementation Schematic of Prey-Localization Model.

This diagram represents our neural-like implementation of the model. The rows of circles correspond with cell-layers. Intercellular connections that end in filled circles are excitatory. Those ending in T's are inhibitory. Only those connections emanating from the center column of cells are shown. The other cells in each layer make similar connections. Although this diagram shows only seven cells in each layer, the actual computer simulation used 41 cells per layer. The accommodation controller is not modeled as a network and is not shown. The imagers, R_L and R_R, supply visual input to the pattern recognizers, T_L and T_R. The cells in the pattern recognizers make simple lateral excitatory connections with their neighbors. As a result, their output is proportional to the visual angle subsumed by a visually-presented object. A more realistic implementation of the model would require more sophistication in these first two layers. The prey selection function is performed by the interaction of the binocular layers, B_L and B_R, the relay layers, N_L and N_R, and the inhibitory layers, U_L and U_R, These layers form a network that will select the region of maximal excitation in the pattern recognizers but with a binocular bias so that both sides will tend to agree on the same visual object.


L
R
R_L
R_R
T_L
T_R
B_L
B_R
N_L
N_R
U_L
U_R
To Accommodation Controller

portrays these layers of cellular units by continuous analytical equations. This convention was adopted for consistency with the description of the "cue interaciton model" of Chapter 3. The reader is referred to Appendix A for a discussion of the development of the mathematical description, and for details of the computer simulation based on this description.

The cell layer representing the imagers simply relays image strength to the pattern recognition layer. The image strength is a function of both the intensity of the incident image and the state of lens accommodation. The effect of lens accommodation on the imagers is represented by varying the intensity of the imager's output signal according to the closeness of the projected point-in-space to accommodation depth. A Gaussian modulating function is used to represent the effect of accommodation and is described in detail in Appendix B. The acuity of accommodation cues decreases with depth. Thus, as with any focusing system, nearby objects are more accurately located in depth than far objects.

The pattern recognition mechanism is not modeled in any detail since we were concerned only with its output. The need for pattern recognition was circumvented by limiting the visual input to a few selected points. The cellular elements of the pattern-recognition layer are modeled as making simple lateral-excitatory interconnections. Therefore, neighboring stimulated points tend to reinforce each other so that the intensity of the recognizer's output in response to an object is a function of both the strength of the signal from the imagers and the retinal angle subsumed by the object. The level of excitation in the pattern recognition layers may be represented in analytical form by the integro-differential equations

$$\mathcal{T}_t\,\dot{T}_L(q,t) = -T_L(q,t) + \int w_t(q - \zeta)\, f_t[T_L(\zeta,t)]\, d\zeta + K_{at}A_L[q,D_a(t)],$$

$$\mathcal{T}_t\,\dot{T}_R(q,t) = -T_R(q,t) + \int w_t(q - \zeta)\, f_t[T_R(\zeta,t)]\, d\zeta + K_{at}A_R(q,D_a(t)]. \tag{4}$$

Profiles of the internal potential in the pattern recognition layers are represented by continuous functions T_L and T_R, with spatial dimension q indicating retinal angle, and t representing time. Internal potential is converted to an external potential (or firing rate) by the saturation-threshold function f_t. The function w_t is a symmetric spread function that describes the extent and strength of the lateral excitatory connections between points in the pattern recognition layers. D_a is the current accommodative state of the lenses, represented on a disparity-based depth scale similar to the scale used in the model of Chapter 3 (see Fig. 14b). It is used, along with retinal angle q, to index maps A_L and A_R of input from the imagers. The pattern recognizers take their input from the imagers through gain K_{at}. The rates of change of potential in layers T_L and T_R are governed by the time constant $\mathcal{T}_t$. Output from the pattern recognizers projects to a layer of binocular cells.

The binocular cell layer is the layer that actually accomplishes the prey selection. The output of this layer projects both to the accommodation controller and to a relay layer that, in turn, sends connections back to both the ipsilateral and contralateral binocular layers. Cells in the binocular layer sum both these ipsilateral and contralateral signals over broad receptive fields (nominally 16.9-degrees). The input to the binocular layers from the pattern-recognition layer is summed over a narrower receptive field. Thus, the binocular cells are excited by a combination of direct stimulation from the pattern recognizers and indirect binocular input. Stimulation from the pattern recognizers assures high angular acuity, and the binocular input biases the selectors so that they tend to select the same visual object. The binocular layer also receives a global inhibitory signal, proportional to the total excitation over the entire binocular layer.

The combination of excitatory receptive fields and global inhibition produces a network that will suppress activity except at that location receiving the maximum stimulation. This network is similar to one proposed by Didday [1970, 1976] (for prey selection in frog) and formalized by Amari and Arbib [1977]. The binocular layers are described analytically by

$$\begin{aligned} \tau_b \dot{B}_L(q,t) &= -B_L(q,t) + \int w_b(q - \zeta)\, f_b[B_L(\zeta,t)]\, d\zeta + I_L(q,t) + I_R(q,t) + \\ &\qquad K_{tb}\, f_t[T_L(q,t)] - K_{ub}\, g[U_L(t)], \\ \tau_b \dot{B}_R(q,t) &= -B_R(q,t) + \int w_b(q - \zeta)\, f_b[B_R(\zeta,t)]\, d\zeta + I_R(q,t) + I_L(q,t) + \\ &\qquad K_{tb}\, f_t[T_R(q,t)] - K_{ub}\, g[U_R(t)]. \end{aligned} \tag{5}$$

Here, B_L and B_R represent internal potential in the binocular layers, f_b is the saturation-threshold function converting internal potential to external potential, w_b is the binocular layer's spread function, and τ_b is its time constant. Inputs I_L and I_R are from the relay layers. The broad spread function associated with these inputs is lumped with the description of the relay layers (see below). The two other inputs are from the pattern recognition layers T_L and T_R through gain K_{tb}, and from the inhibitory layers U_L and U_R through gain K_{ub}.

The relay layer receives its input from the binocular cells and has two distinct outputs. One set of outputs goes to the inhibitory layer, and the other projects to both the ipsilateral and contralateral binocular cell layers. Projections to the inhibitory layer provide that layer with a signal modulated by total activity across the entire binocular layer. Projections to the binocular layers provide the cross-coupling pathways between these two layers. Since the function of the relay layers is simply to pass information to other layers, their transient characteristics can be lumped with those of the other layers. Thus, the equations used to describe the relay layers are equilibrium equations rather than differential equations, and are given by

$$I_L(q,t) = \int w_i(q - \zeta)\, f_b[B_L(\zeta,t)]\, d\zeta,$$

$$I_R(q,t) = \int w_i(q - \zeta)\, f_b[B_R(\zeta,t)]\, d\zeta. \qquad (6)$$

In these equations I_L and I_R represent the potential transmitted by the relay layers to the binocular and inhibitory layers. The function w_i provides the broad spread of relay potential as received by the binocular layers. This spread function was applied directly to the relay layers and not to their inputs to the binocular layers, in order to give the experimenter direct access to a potential representing the relay's inputs to the binocular layers.

The inhibitory layers are single units that simply integrate activity across the entire ipsilateral relay layer. Since the dynamics of the relay layers were not modeled in the simulation, this integration was actually performed over the ipsilateral binocular layer. Thus, the inhibitory layers are described by

$$\tau_u \dot{U}_L(t) = -U_L(t) + K_{bu} \int f_b[B_L(\zeta,t)] \, d\zeta,$$

$$\tau_u \dot{U}_R(t) = -U_R(t) + K_{bu} \int f_b[B_R(\zeta,t)] \, d\zeta, \qquad (7)$$

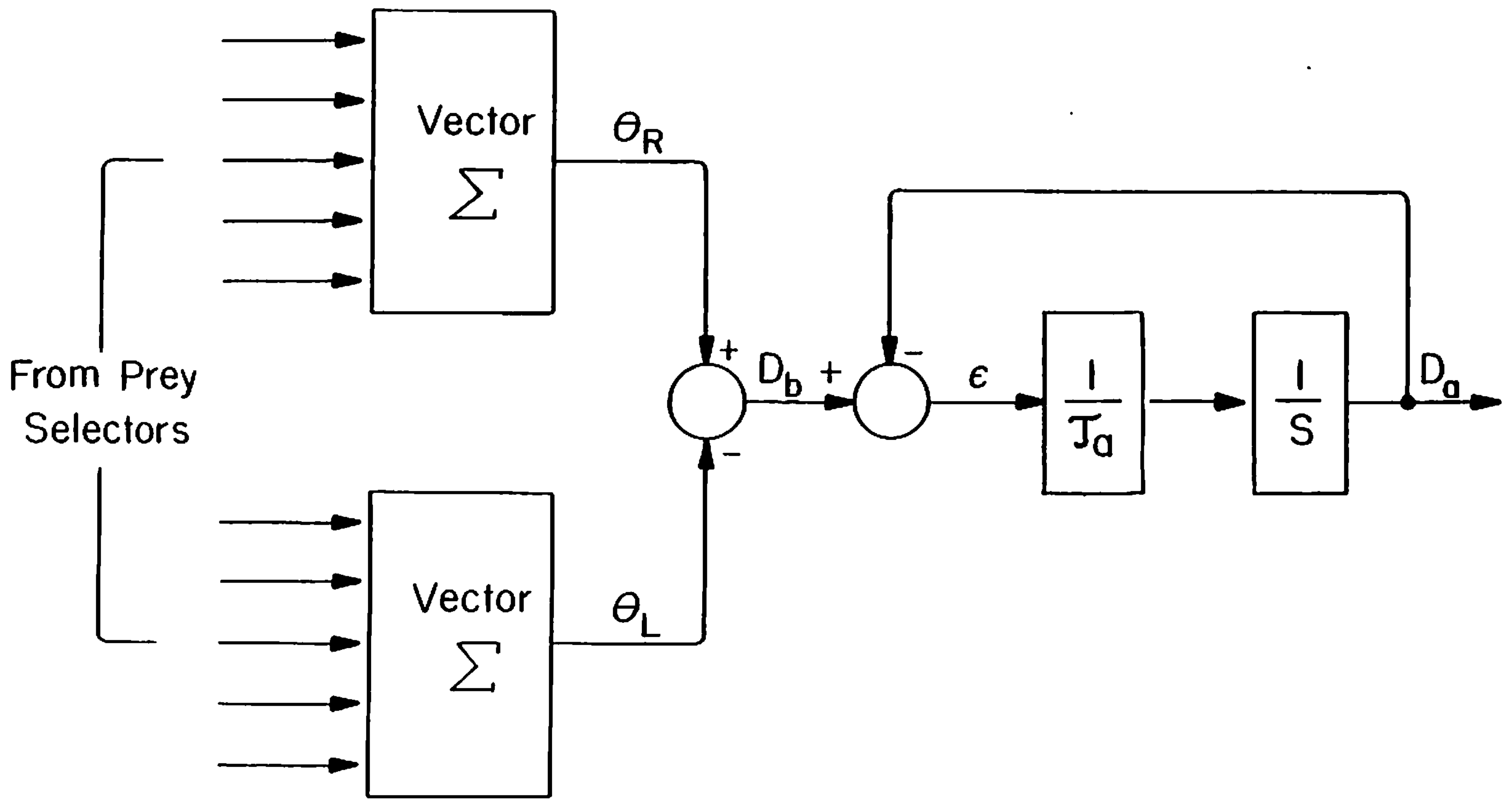
Here, U_L and U_R represent the internal potentials in the inhibitory layers, and τ_u is the layers' time constant. The gain K_{bu} is applied to the input from the binocular layers.

The mechanism for adjusting lens accommodative state, based on the output of the prey selectors, is depicted in Fig. 31. The first process carried out in this mechanism is to identify a pair of image coordinates by determining the center of gravity of the activity pattern on each of the two prey selectors. In this process, the output from each point in the binocular cell layer is treated as a vector in a radial coordinate system, with its angular component representing retinal position and its radial component encoding the likelihood that there is a prey at that position. The vectors from each binocular layer are summed to produce a resultant whose angular component is the retinal position corresponding with the center of gravity, or locus of average excitation, on that side of the visual system.[1] The loci of average excitation θ_L and θ_R in the binocular layers are determined by the formulae

[1] The notion, employed here, of motor circuits computing a vector sum of their input in order to control output was originally advanced by Pitts and McCulloch [1947]. Their hypothesis has found recent confirmation in an analysis of monkey arm movements by Georgopolis et al. [1983], and by McIlwain [1982] in his investigation of cat superior colliculus.

Fig. 31 - The Accommodation Controller.

The accommodation controller is modeled as a lumped-parameter system. Its input is the pattern of excitation in the prey selectors. The controller treats these inputs as two sets of vectors, with each vector associating a scalar weight with visual-angle on the image surface. By computing the vector-sum of these inputs the controller determines an attention-angle on each image, θ_R for the right-eyed image and θ_L for the left-eyed image. The difference or disparity, D_b, between the attention angles is used as a measure of depth. This depth is compared with the current depth-setting, D_a, of the controller to compute an error signal ϵ. The regulator in the controller is modeled as a linear first-order lag-circuit with time-constant T_a. This time-constant is meant to incorporate any delay in the controller and the lens-focusing mechanism.


From Prey Selectors
Vector $\sum$
Vector $\sum$
θ_R
θ_L
+
−
D_b
+
−
ϵ
$\frac{1}{J_a}$
$\frac{1}{S}$
D_a

$$\theta_L(t) = \tan^{-1} \frac{\int f_b[B_L(\zeta,t)] \sin(\zeta / a)\, d\zeta}{\int f_b[B_L(\zeta,t)] \cos(\zeta / a)\, d\zeta},$$

$$\theta_R(t) = \tan^{-1} \frac{\int f_b[B_R(\zeta,t)] \sin(\zeta / a)\, d\zeta}{\int f_b[B_R(\zeta,t)] \cos(\zeta / a)\, d\zeta},$$
(8)

where the scalar parameter a is simply a conversion factor between visual angle and retinal position. The two retinal angles θ_L and θ_R determined by this vector summation are subtracted to give disparity and thus a measure of depth, according to the formula

$$D_d(t) = a\, [\theta_R(t) - \theta_L(t)], \tag{9}$$

This depth estimate D_d is the setpoint for the controller. The controller's error signal is computed by subtracting an estimate of current lens accommodative state D_a from this setpoint. The controller's transfer function is simply a first-order lag with time constant T_a. It is governed by the differential equation

$$T_a\, \dot{D}_a(t) = -D_a(t) + D_d(t). \tag{10}$$

This scheme for controlling the lens is similar to the efferent-copy scheme that Robinson [1981] utilizes in his model of collicular control of eye movement.

The decision to use an efferent copy, rather than proprioceptive feedback, to provide an estimate of the accommodative state of the lens, was not arbitrary. Toads whose lens accommodation muscles are treated with the muscle relaxor Atropine tend to undershoot prey. When these muscles are treated with the muscle contractor Miotic they tend to overshoot [Jordan et al., 1980]. When the toad's accommodation muscles are relaxed their lenses are adjusted for far vision. Contraction adjusts the lenses for near vision. Thus, if proprioceptive cues were used to estimate lens position Atropine treatment would cause overshooting and Miotic would cause undershooting. If, on the other hand, an efferent copy were used then the observed effect would be the expected one.

The accommodation control algorithm is designed to work perfectly in the case when both prey selectors have selected the same visual object. If this has occurred, then the only retinal position excited in each binocular layer will be the one corresponding with the position of the selected prey, and the retinal position from the vector summation will coincide with the position of the prey. Depth can be directly derived from two such retinal positions by a process of triangulation (see Appendix B).

However, before the prey selectors have produced such a refined output the pattern of excitation over the binocular layers is usually more complex. In this case the desired depth, computed from the two centers of gravity, will probably not correspond with the actual depth of the true prey object. It will only be an approximation. This approximate depth will generally help to improve focus and, in turn, assist the prey selectors in converging on a single position.

The accommodation controller also contains a refinement that is not depicted in Fig. 31. The magnitude of each vector sum is compared with a small threshold. If either magnitude is below this threshold then the output of the corresponding prey selector is assumed to be too weak to be used to specify a retinal angle. In this case the desired depth setting of the controller is set to a neutral or rest position.

Graphics displays

The two types of display used to show model output are depicted in Fig. 32. Fig. 32a is a schematic top-view of a 40 × 40 cm. square arena. Grid markings are spaced at 10 cm. Below the arena is an icon representing a frog or toad. Filled discs represent the eyes. The filled rectangle in the grid represents a prey, and the lines projecting into the grid from the centers of the eyes indicate the attention angles calculated by the model. The intersection point of these lines corresponds with the depth being estimated by the binocular system. In this simple test case the model has correctly located the prey. Fig. 32b depicts the internal state of the model at the same time. The square planes at the top of the figure are maps relating imager output to the possible accommodative states of the lenses. Their coordinates are visual angle, along the horizontal axis, and disparity (a non-linear representation of depth), along the vertical axis. The circles within these squares indicate, by their size, the strength of the retinal signal that would be produced by accommodating the lens to the depth indicated on the vertical axis. The narrow horizontal rectangles superimposed on these maps indicate the current accommodative state of the lenses. Thus, the area abutting these rectangles represents the current state of the imagers. The three pairs of line graphs below these maps indicate the levels of excitation in the three other layers of the model. The top curves show activity in the pattern recognizers, the middle curves show the strength of the feedback to the binocular layers from the relay layers, and the bottom curves show activity in the binocular layers.

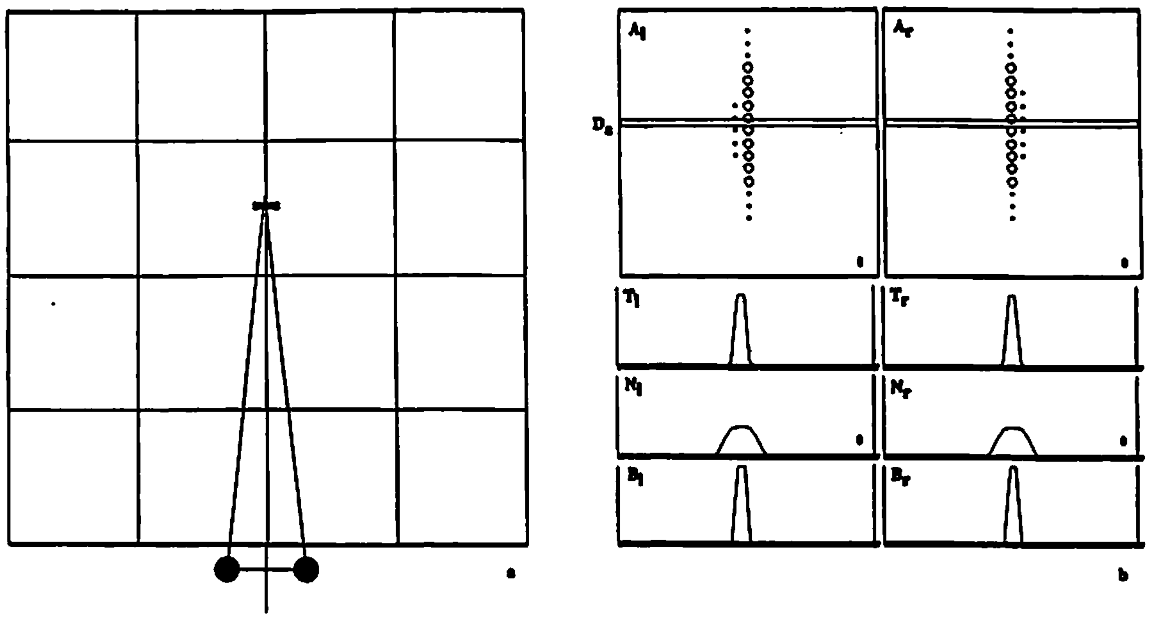


Fig. 32 - Displays Available from the Computer Simulation.

(a) This display shows both a top-view of the scene being processed and the resulting depth-estimate. The square grid represents a 40 cm. × 40 cm. arena. Each grid element is 10 cm. on a side. The simulated frog or toad is indicated by the icon below the arena. Filled circles on the cross-bar indicate eye position. The filled rectangle within the arena represents a prey object. The lines emanating from the eyes and converging on the prey indicate the current attention-angles of the two eyes. The point-of-intersection of these lines determines the model's current estimate of the spatial-location of the prey.

(b) This display shows the model's internal state. Squares A_L and A_R represent the full range of possible accommodative states for the left and right eyes. Their vertical coordinates are depth (on a non-linear disparity scale), D, and their horizontal coordinates are retinal angle, θ. The long narrow rectangles superimposed on these squares indicate the current accommodative state, D_a, of the lenses. The circles within the squares indicate both retinal angle and strength (size-encoded) of a visual stimulus for each accommodative state. The three line-graphs below these squares show activity in the pattern recognizers, T_L and T_R, the relay layers, N_L and N_R, and the binocular prey selector layers, B_L and B_R. Their vertical axes indicate level of excitation and their horizontal axes retinal angle.

Any binocular imaging system is subject to making errors in binocular correspondence under conditions that vary according to the algorithm being employed. The prey-localization model is no exception to this rule. Since this model uses lens accommodation to help disambiguate binocular matching, it is particularly subject to errors when presented with a visual configuration that is symmetric about the body axis of the simulated animal. Several of our experiments were directed at determining parameter settings that would minimize binocular mismatch under these conditions.

Depth estimate categories

In order to provide a consistent means for measuring model performance when it was tested with symmetrically balanced visual input we defined a set of categories of depth estimates. Fig. 33 depicts these categories. Since each side of the visual system selects a particular visual angle, the depth estimate is the point of intersection of rays traced from the selected retinal point through the pupil. Category a) of Fig. 33 consists of the locus of points in space corresponding to a "hit" or a correct estimate. This will occur when both rays pass through the same target. Category b) depth estimation errors represent the usual binocular or "crossed-ghost" error, made when each eye selects the target in its contralateral field. Category c) errors, that we call the "uncrossed-ghost" error, are those made when each eye selects the object in its ipsilateral field. As long as the objects in the field are separated by more than the inter-pupillary distance this kind of error will result in depth estimates that lie behind the eyes. Categories d) through f) correspond with cases when one or both eyes select a point between the two visual

Fig. 33 - Categories of Depth Estimates.

A depth-resolving system based on binocularity will make a correct depth-estimate when like points on both retinal images are paired but will make errors when points are incorrectly paired. The categories of depth-estimate shown in this figure were used in evaluating the model's performance when confronted with a visually balanced scene containing two prey. In all figures the dots represent pupil position, the filled rectangles represent prey, and the area between the two rays passing through the pupils represents a range of angular orientations. The hatched area represents the locus of depth-estimates that would result from binocularly pairing angular orientations from within the indicated ranges. The possible categories of depth-estimate are:

a) hit - both eyes converge on the same object,

b) crossed-ghost - the binocular mismatch error usually described in the literature. This error occurs when each eye selects the object in the contralateral visual field,

c) uncrossed-ghost - another true mismatch error in which each eye selects the object in the ipsilateral visual field,

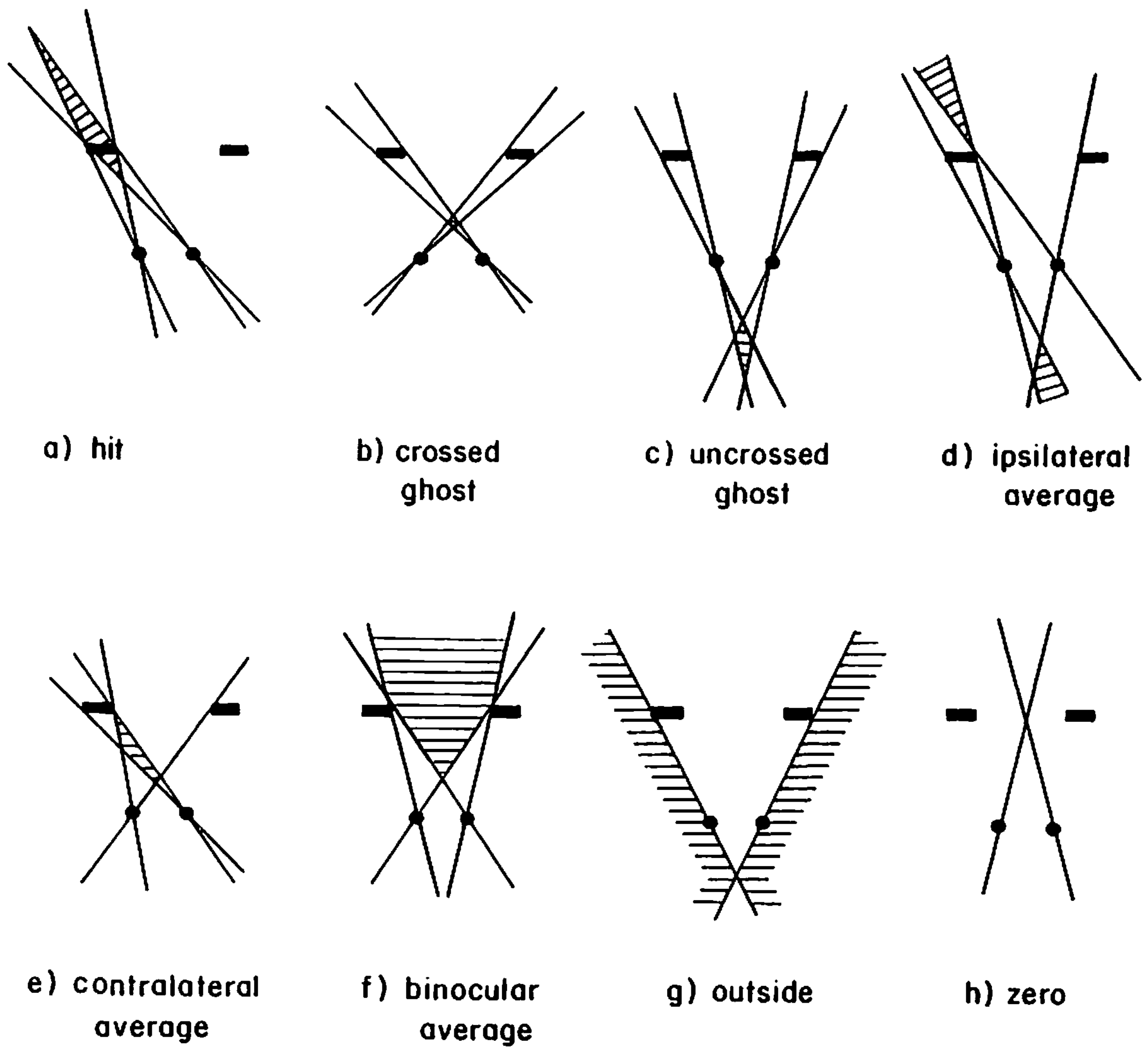
d) ipsilateral-average - one eye selects the ipsilateral object but the other is directed towards a point between the two objects,

e) contralateral-average - one eye selects the contralateral object but the other is directed towards a point between the two objects,

f) binocular-average - both eyes select a point between the two objects,

g) outside - either one or both eyes selects a point peripheral to the two objects,

h) zero - when neither eye selects a point there is no depth estimate and the system can be thought of as being directed at some neutral or resting position.


a) hit
b) crossed ghost
c) uncrossed ghost
d) ipsilateral average
e) contralateral average
f) binocular average
g) outside
h) zero

stimuli.[2] We have labeled these categories the d) "ipsilateral-average" error, the e) "contralateral-average" error, and the f) "binocular-average" error. Category g) or "outside" errors are made when at least one eye selects a point that does not lie either on or between the two stimuli. Finally, for completeness, we include the "zero" error that occurs when neither eye selects any point, leaving the animal's attention directed at a neutral or "relaxed" point.

Visual input

Experiments to evaluate model performance with symmetrical input consisted of a series of 21 runs. The visual input supplied for these runs was two identical objects placed symmetrically about the midline and at equal distances from the eyes. To generate the 21 different cases the objects were placed at 10, 20, and 30 cm. from each other and at 7, 12, 17, 22, 27, 32, and 37 cm. from the interpupillary line. Results from these runs were tabulated in the form of histograms, with one bin for each of the 8 categories of depth-estimate. When runs were made with the intent of performing a sensitivity analysis the model was run on all 21 cases for each of 6 values of the parameter being studied. Thus, histograms contain 6 bins for each category for a total of 48 bins.

[2] Such off-target selection could easily occur in the brain if areas of excitation spreading simultaneously from two stimulated loci interact to create a false peak of excitation between the original stimulus sites.

Results

Experiments with single prey stimuli

Experiments with a single prey-stimulus were done to verify that the model would function as expected, to tune the model's parameters, and to test the effects of simulated lenses and prisms in the visual-field. Fig. 34 shows typical results from these experiments.

When lenses and prisms were not used the model converged rapidly and accurately upon a single stimulus wherever it was placed in the binocular field. Fig. 34a depicts three such experiments. The reduction in accuracy for the most distant prey is within the bounds determined by the grain-size of the simulation.

When only a single prey was present, the presence of simulated lenses did not affect accuracy but did increase convergence time. With simulated lenses set to produce a 30 percent shift in the disparity coordinate of the peak of the lens accommodation curve the average convergence time for 8 different single-prey presentations was 1.8 times that obtained without lenses. Fig. 34b shows typical results for the trials with lenses. Accuracy is not affected by the lenses since accommodation is not used as a depth estimator but only to disambiguate binocular matching. With only a single stimulus there is no ambiguity. The convergence time increased since the defocusing of the lenses reduced the level of excitation in the pattern recognizers and thus acted to reduce the total gain of the system.

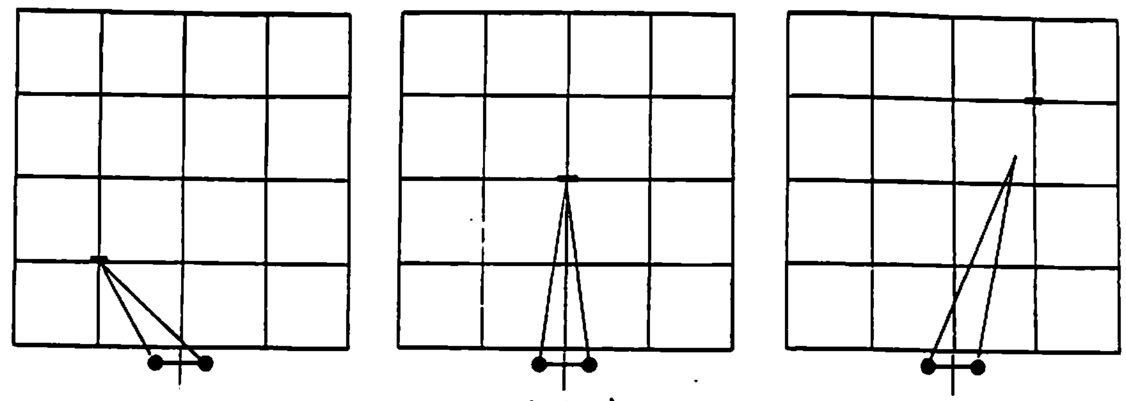
Because the depth estimate computed by the model is based solely on binocular parallax, placing base-out prisms in front of the eyes caused undershooting errors. Fig. 34c illustrates this effect. Here, simulated base-out prisms, set to

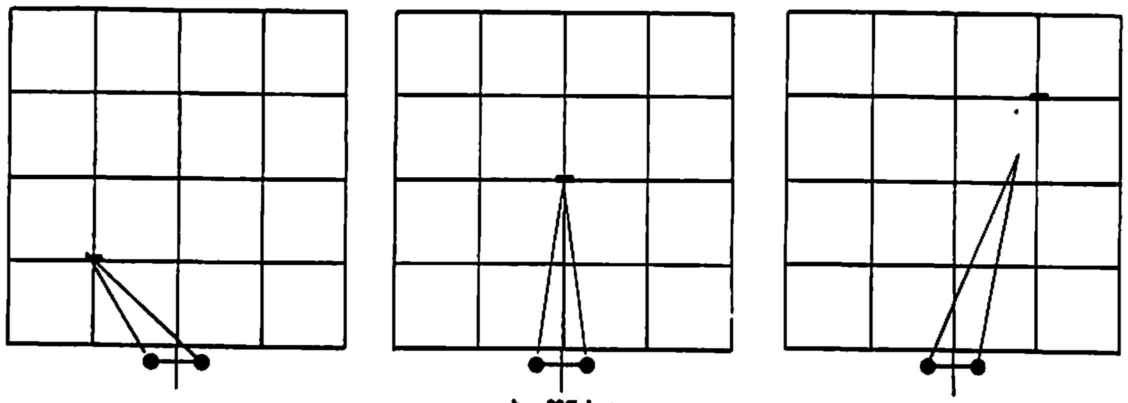
Fig. 34 - Sample Results for a Single Prey.

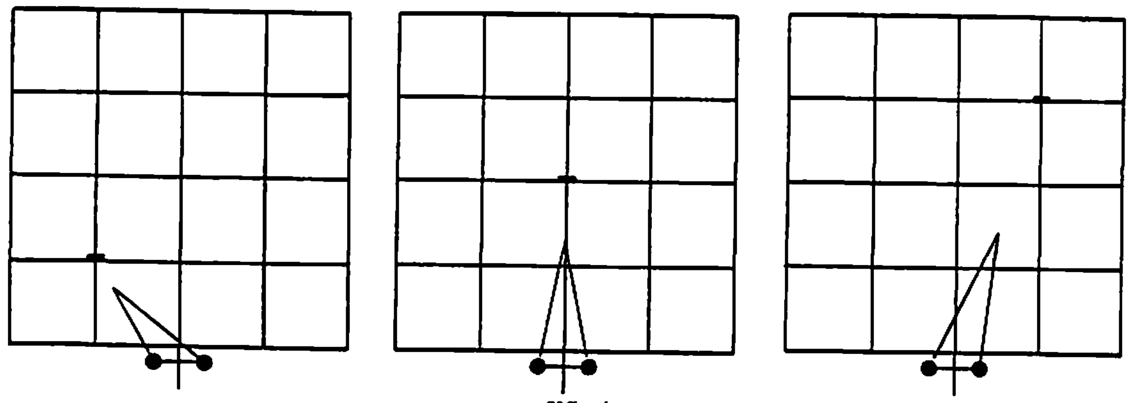
a) The model correctly localizes a single prey at various positions.

b) Simulated lenses producing a 30% shift in accommodative depth estimate have no effect on single-prey localization.

c) Simulated prisms, producing a 30% shift in binocular parallax, cause undershooting errors.


a – normal


b – 30% lens


c – 30% prism

produce a 30 percent shift in the disparity coordinate, caused the expected underestimation of prey distance. Again, the discrepancy between the binocular depth estimate and the lens accommodative state reduced the system gain and resulted in a convergence time 1.8 times greater than that obtained without prisms.

Experiments with multiple prey stimuli

Visual configurations consisting of more than one stimulus present a correspondence problem that any binocular depth resolution system must be equipped to solve. In our model, lens accommodation provides the additional information necessary to assure correct correspondence. Lens accommodation will be most effective in disambiguating binocular matching when the visual stimuli are at different depths. To confirm that our model is successful in correctly interpreting this type of configuration we examined its response to several two-prey configurations where the prey were placed at different distances from the simulated frog. Fig. 35 depicts typical results from a series of such experiments. The model converged upon a correct depth estimation in all but three out of 120 trials. Of the three incorrect estimates two were "uncrossed-ghost" errors and the third was a "zero" error. In all correct cases the selected prey was the one closer to the animal. The simulation was biased towards the nearer of the two prey by the lateral facilitation in the pattern recognizers. This caused their output to be modulated by the retinal angle subsumed by the visual stimulus. Thus, not only are the mechanisms envisioned in the model successful in overcoming ambiguity in binocular matching but they faithfully mimic the demonstrated preference of frogs and toads for the nearer of two prey.

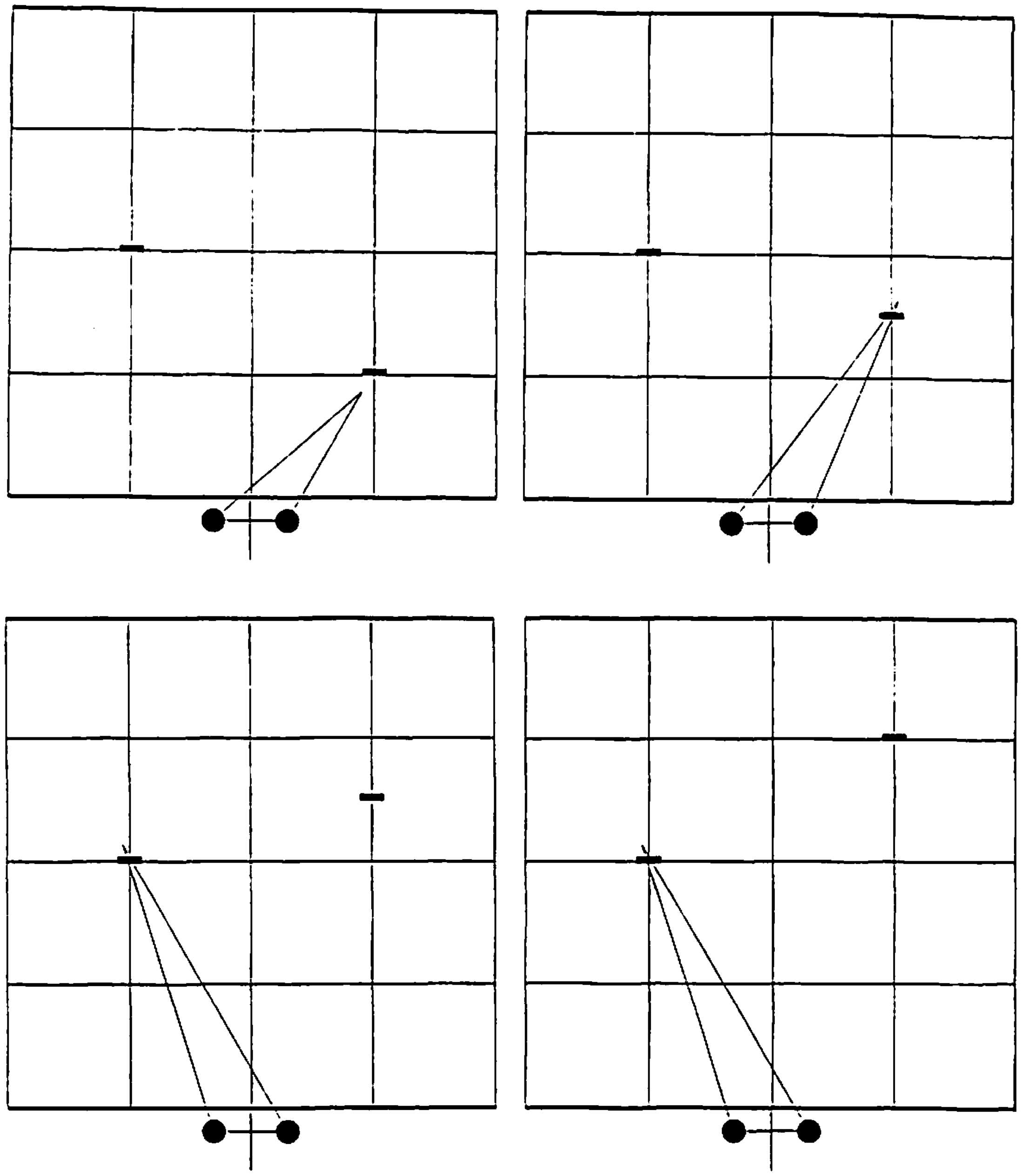


Fig. 35 - Results with Two Prey at Different Depths.

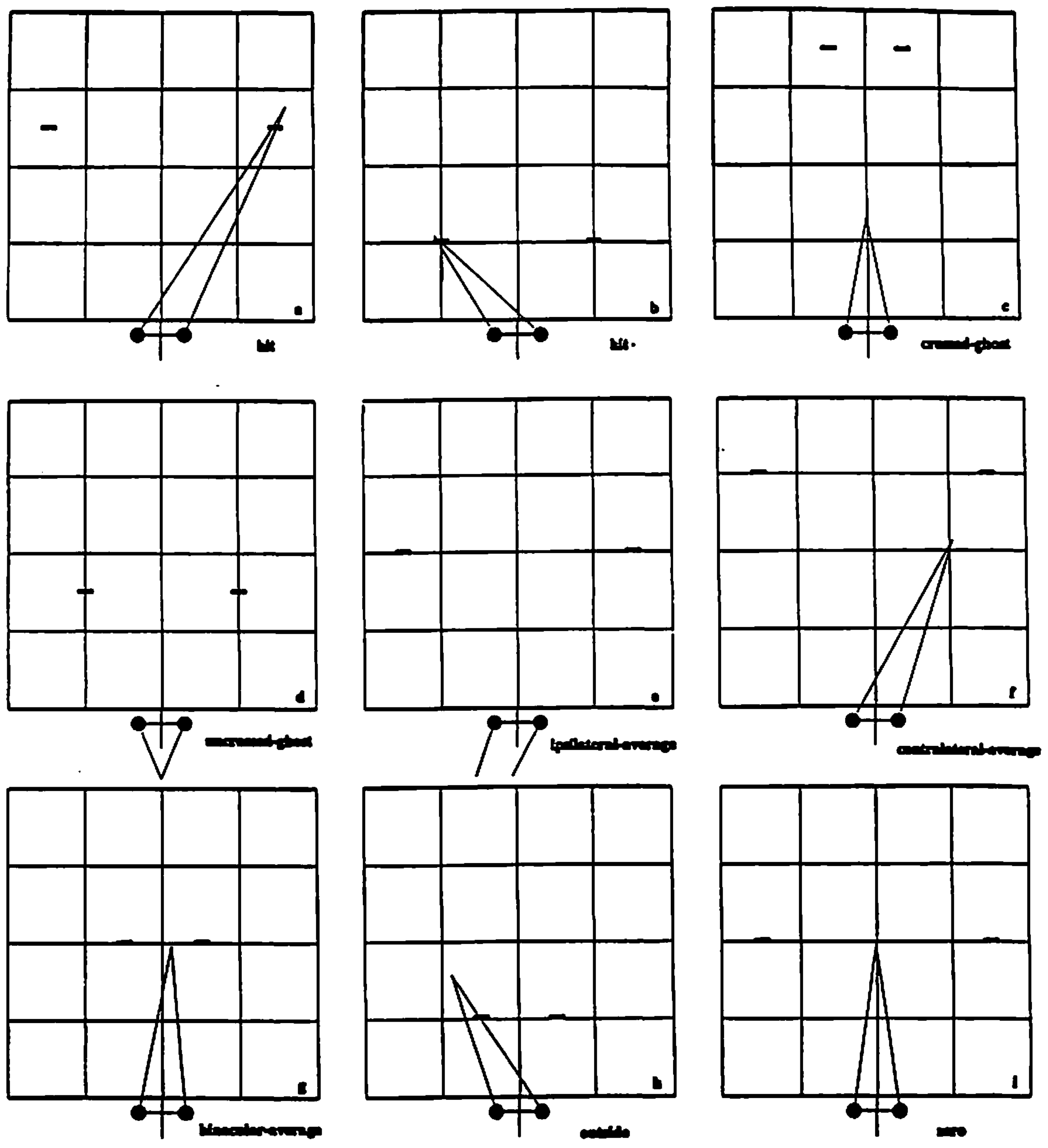
When two prey are presented simultaneously but at different depths the model correctly localizes the nearer prey. a) through d) show typical results for four such configurations.

A much more difficult problem for our model arises when two prey are placed at identical depths. In this case the use of accommodation cues will not assist in disambiguating the binocular matching problem. However, the crossed pathways between the prey selectors are effective, in most cases, in resolving the ambiguity. Fig. 36 shows the range of results obtained during experiments with two prey presented symmetrically about the animal's midline. With the model tuned for optimal performance it produced correct depth estimates for nearly 80 percent of the test cases. Figs. 36a and 36b show two representative cases in which a correct depth estimate was made.

Under certain circumstances binocular cross-coupling is not enough to assure a correct match. The most difficult situation is when the two prey are close enough together so that the visual angle separating them is less than the receptive-field size of the binocular crossed connections in the prey selectors. In this case two distinct kinds of errors can be made. First, the model may make a "ghost" error. If each eye selects the object in its contralateral visual field, then a "ghost" prey will be selected such as that shown in Fig. 36c. This "ghost" will be between the actual prey and much closer to the animal than either of them. If each eye selects the ipsilateral object then the "ghost" will appear to be behind the eyes, as shown in Fig. 36d. Second, the model may make one of the three "average" errors. In these cases the model may choose an "average" prey located at a position between the two true prey. The "ipsilateral average" error is shown in Fig. 36e, the "contralateral average" error in Fig. 36f, and the "binocular average" error in Fig. 36g. Besides these "ghost" and "average" errors it is possible to detune the model

Fig. 36 - Results with Two Prey at the Same Depth.

When two prey are positioned symmetrically about the midline, depth-cues from accommodation are nullified. However, because of the binocular cross-coupling between the prey selectors the model is usually able to converge upon a correct depth-estimate. a) and b) show two such cases. Under various tunings the model can produce depth-estimates that fall into each of our other seven categories. c) through i) are representative examples of each of these erroneous responses.


a
hit
b
hit ·
c
crossed-ghost
d
uncrossed-ghost
e
ipsilateral-average
f
contralateral-average
g
binocular-average
h
outside
i
zero

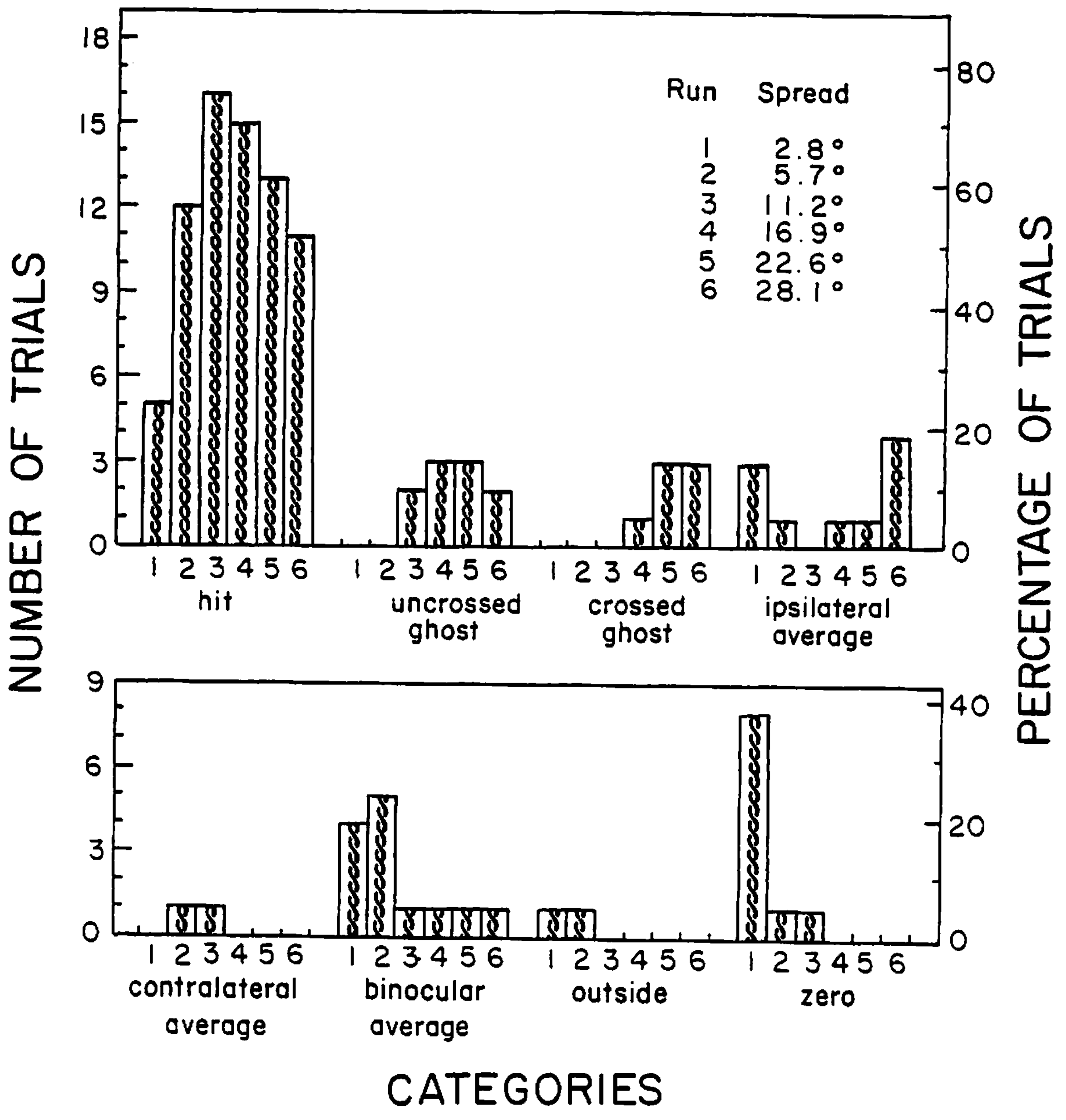
so that it will locate a prey peripheral to either of the true prey as in Fig. 36h or to ignore both prey as in Fig 36i.

The parameters that appeared to have the greatest effect on the response of the model to symmetric presentations were 1) the receptive field width in the binocular layer, 2) the net input gain in the binocular layer, and 3) the time-constant of the accommodation controller. A series of three experiments was run to evaluate the effect of varying each of these parameters. For each experiment the standard test-suite of 21 symmetrical configurations was tried against 6 different values of the parameter being evaluated. Results were tabulated in the form of histograms.

The results of the experiment designed to examine the effect of binocular receptive field width are tabulated in Fig. 37. The six runs in the experiment were done with receptive field width set at 2.8, 5.7, 11.2, 16.9, 22.6, and 28.1 degrees. The net gain of the binocular layer's excitatory spread function was held constant by reducing its height in proportion to its increase in width. With the spread set at its lowest value (2.8 degrees) binocular parallax is large enough for most configurations that the binocular fields do not overlap and thus the number of correct matches is only 25 percent. For all other configurations the binocular system is essentially disfunctional and the system either does not respond (zero error) or responds with one of the "average" errors. With the optimal setting between 11.2 and 16.9 degrees responses are nearly 80 percent correct. The only error occuring in more than one case is the "uncrossed-ghost" error. Increasing the binocular spread to 28.1 degrees results in an increase in both "crossed-ghost" and "ipsilateral-average" errors.

Fig. 37 - Model Performance vs. Binocular Spread.

Histograms show the number of estimates that fell in each of the eight depth-estimate categories for each of six values of the binocular-spread parameter. Net binocular gain was held constant and the standard suite of 21 symmetrical presentations was used for each of the six runs. The binocular-spread parameter governs the extent of the excitatory receptive field in the binocular layer. This ERF size was varied from 2.8 degrees in run 1 to 28.1 degrees in run 6.


NUMBER OF TRIALS
PERCENTAGE OF TRIALS
Run Spread
1 2.8°
2 5.7°
3 11.2°
4 16.9°
5 22.6°
6 28.1°
hit
uncrossed ghost
crossed ghost
ipsilateral average
contralateral average
binocular average
outside
zero
CATEGORIES

Adjusting the net input gain in the binocular layers has a dramatic effect on the number of correct localizations vs. the number of "average" errors. Fig. 38 shows model performance as the binocular gain is varied from 2 to 167 percent of its nominal setting (run 4). With the binocular gain very low there is a preponderence of "binocular-average" errors. This is because the input to the prey selectors is too weak to allow the selection process to isolate a single prey. The vector summation performed in the accommodation system then simply chooses an average point of stimulation. As binocular gain is increased to near-nominal the prey selectors begin to operate properly and the number of "average" errors drops off dramatically. Finally, with the binocular gain increased above nominal the number of "average" errors increases again. The reason for this increase is that excitation is so high in the binocular layers that there is significant spread to lateral positions. Thus, the prey selectors cannot successfully select a single visual angle but, instead, select a broader region. This also accounts for the fact that with high binocular gain there are a significant number of cases of both "ipsilateral" and "contralateral average" errors. In both these cases one of the eyes selects a prey whereas the other selects an average angular location.

The time constant T_a of the accommodation controller had a less dramatic but noticeable effect upon model performance. Fig. 39 shows the results of experiments investigating the effect of varying this time constant. Run 4 was done with the time-constant set at its nominal value. Although the value of the time constant does not have a large effect on the total number of "hits" vs. incorrect depth estimates there is an effect upon the types of errors made. With a small time

Fig. 38 • Model Performance vs. Binocular Gain.

Histograms show the number of estimates that fell in each of the eight categories for each of six values of the binocular-layer's input gain. Binocular spread was held constant at 16.9 degrees and the standard suite of 21 symmetrical presentations was used for the six runs.

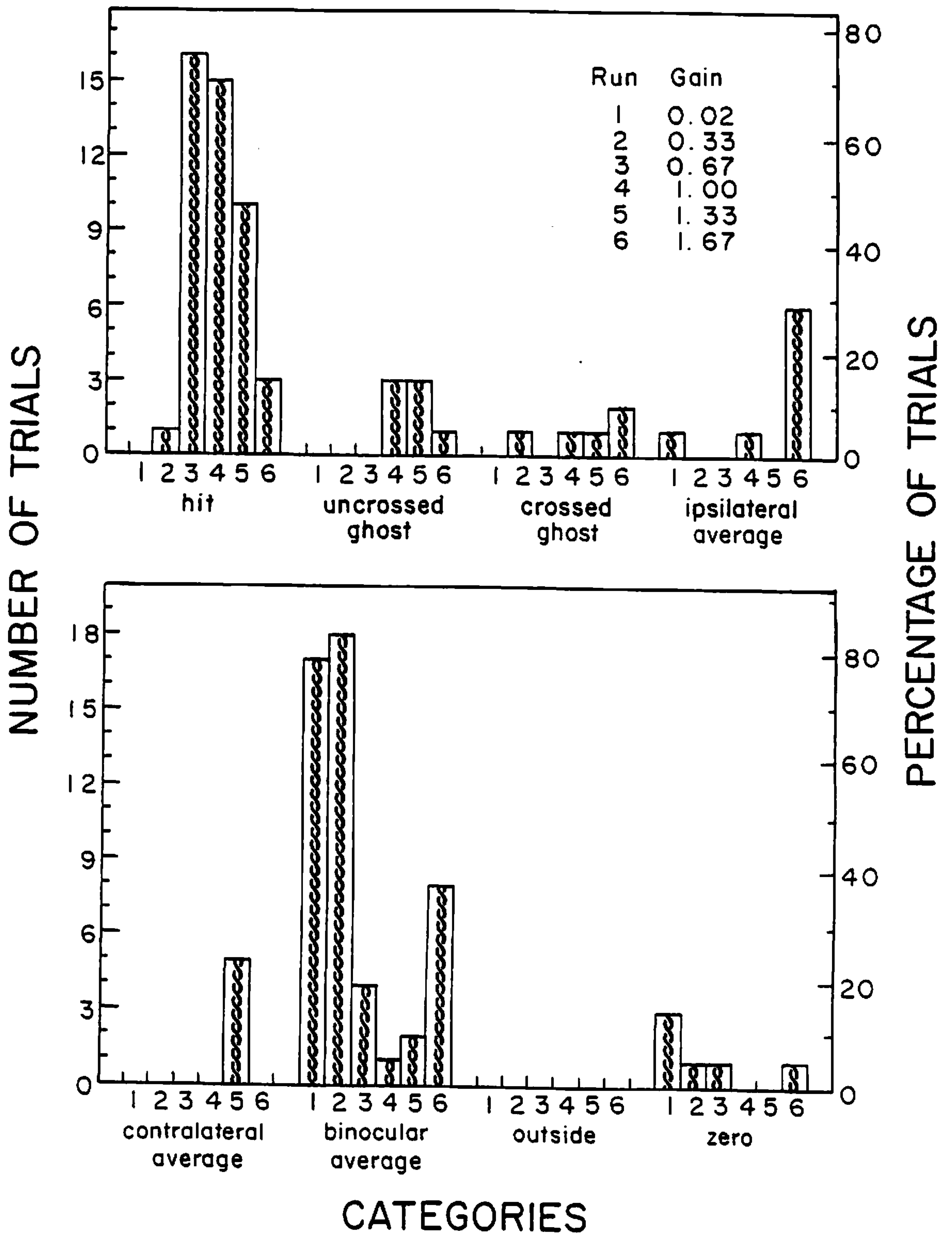
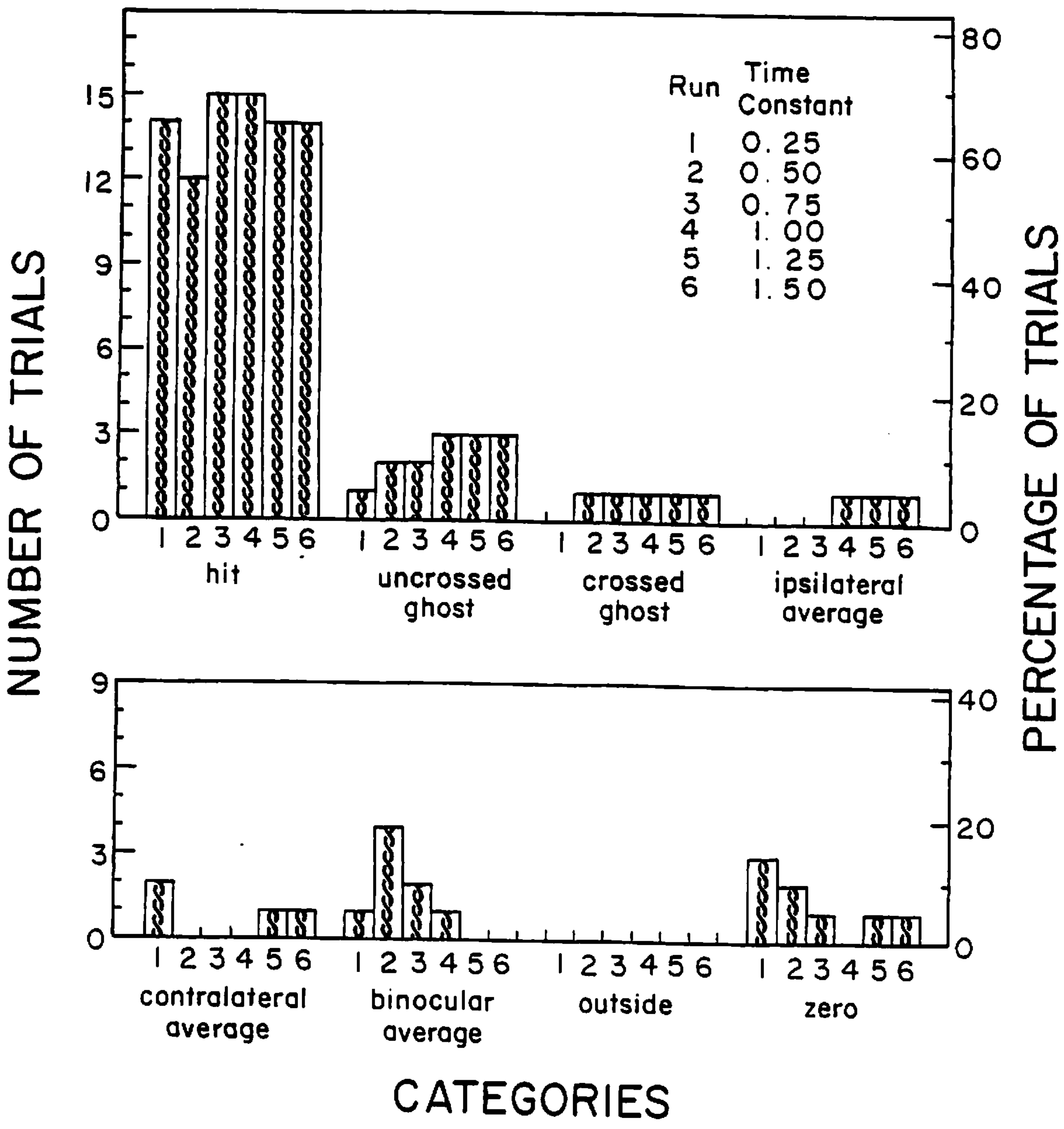

Run Gain
1 0.02
2 0.33
3 0.67
4 1.00
5 1.33
6 1.67
NUMBER OF TRIALS
PERCENTAGE OF TRIALS
0 3 6 9 12 15
0 20 40 60 80
1 2 3 4 5 6
hit
uncrossed ghost
crossed ghost
ipsilateral average
0 3 6 9 12 15 18
0 20 40 60 80
contralateral average
binocular average
outside
zero
CATEGORIES

Fig. 39 - Model Performance vs. Accommodation Time Constant.

Histograms show the number of estimates that fell in each of the eight categories for each of six values of the time-constant in the accommodation controller. The standard suite of 21 symmetrical presentations was used for the six runs.



NUMBER OF TRIALS
PERCENTAGE OF TRIALS
Run
Time Constant
1 0.25
2 0.50
3 0.75
4 1.00
5 1.25
6 1.50
hit
uncrossed ghost
crossed ghost
ipsilateral average
contralateral average
binocular average
outside
zero
CATEGORIES

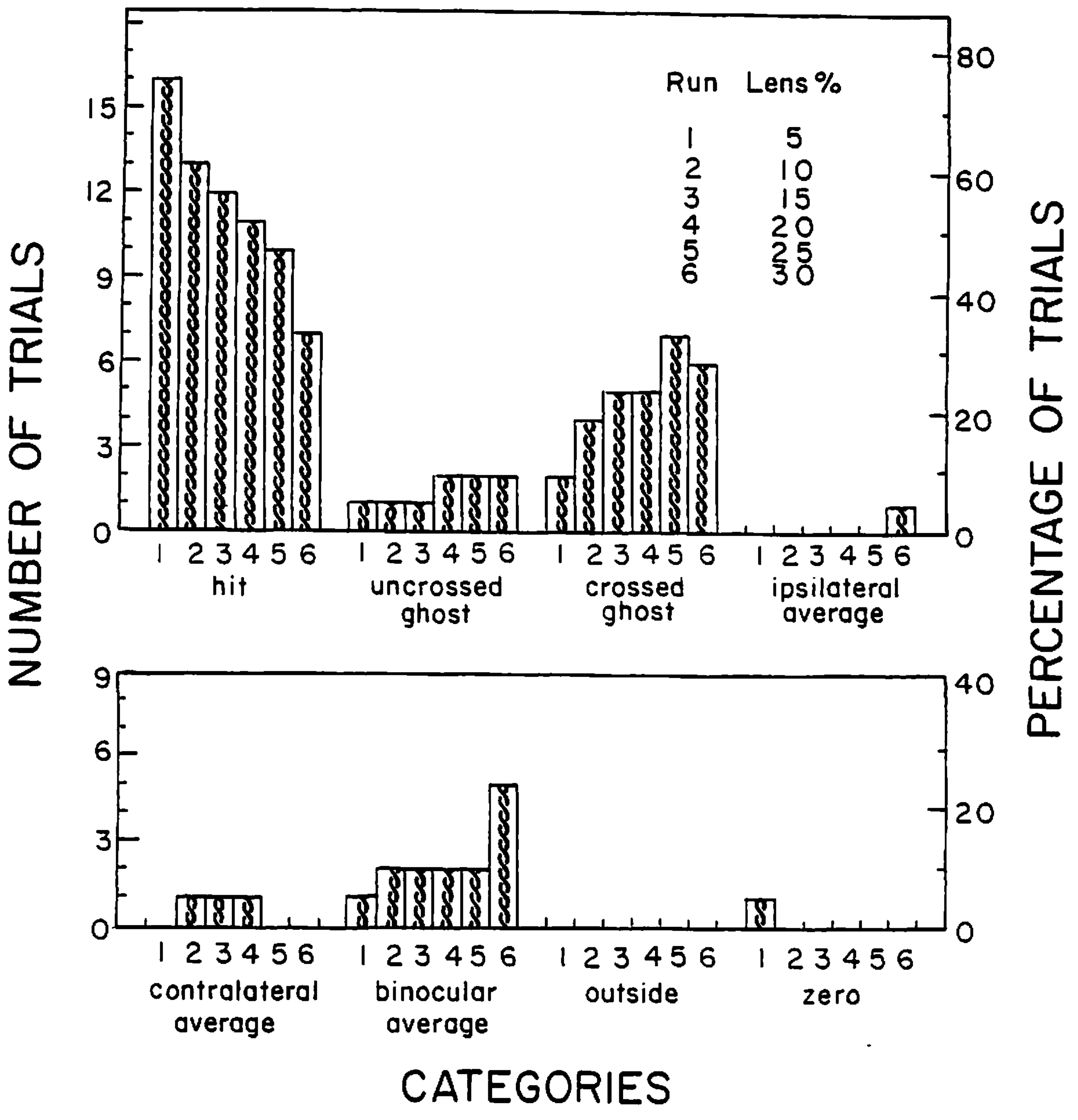
constant (25-50 percent of nominal) most of the errors are "zero" or "average" errors. However, as the time constant is increased to 150 percent nominal, the number of "uncrossed-ghost" errors increases. The high number of "zero" and "average" errors at low time constants is due to the fact that the lenses are accommodated to a particular depth before the prey selectors have converged. Thus, accommodation depth tends to be inaccurate. The resulting defocus has an effect similar to that obtained by lowering the binocular gain.

The effect of lenses on two-prey experiments

Collett and Udin [1983] were able to produce snapping errors in both normal and lesioned toads by placing concave lenses in front of their eyes. We were able to partially reproduce this effect with our model. Fig. 40 shows model performance on the symmetrical test suite when simulated concave lenses of various strengths were interposed between the eyes and the test-scene. As lens strength was increased, the number of errors made by the model also increased. The errors fell mostly into the "crossed-ghost" and "binocular-average" categories. Collett and Udin's description indicates that actual toads with lenses affixed in front of their eyes exhibit "binocular-average" errors as well as a marked tendency to avoid snapping.

Fig. 40 - Model Performance with Interposed Lenses.

Histograms show the number of estimates that fell in each of the eight categories for each of six strengths of simulated concave lenses placed in front of the eyes. The standard suite of 21 symmetrical presentations was used for the six runs.


NUMBER OF TRIALS
PERCENTAGE OF TRIALS
Run Lens %
1 5
2 10
3 15
4 20
5 25
6 30
hit
uncrossed ghost
crossed ghost
ipsilateral average
contralateral average
binocular average
outside
zero
CATEGORIES

Discussion

We have demonstrated a model of depth perception whose development was guided by consideration of the prey localization process in frogs and toads. The model does not attempt to build a global depth map of a visual scene but rather locates a single point in space. It utilizes depth from binocular matching to guide lens accommodation that, in turn, assists in disambiguating binocular matching.

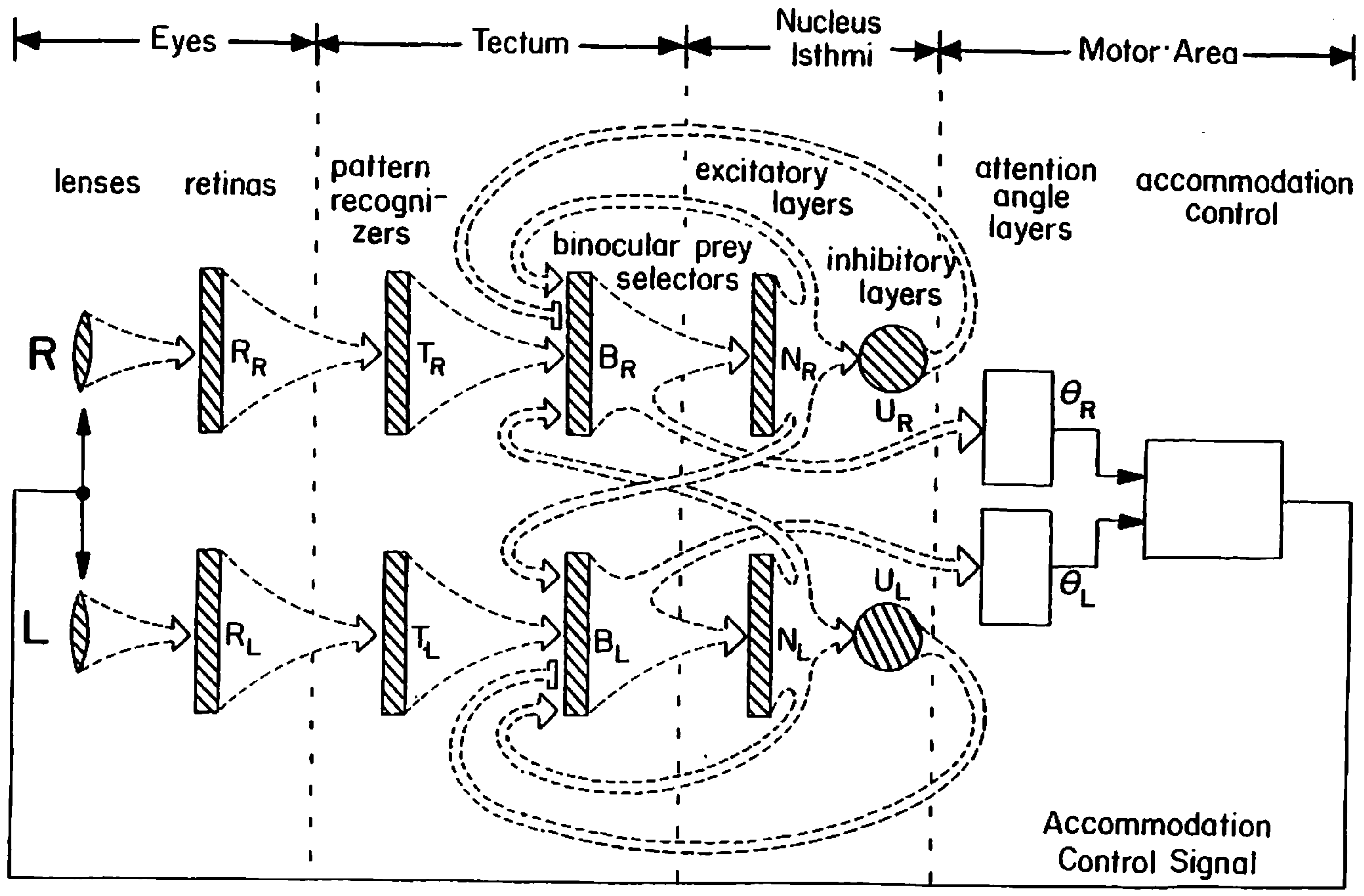
A possible neural realization of the model

The structure of the model was influenced quite strongly by our study of the stages of the visual system of frogs and toads. Our proposal for how the model's structure might be implemented in this system is shown in Fig. 41. In this figure the two sides of the visual system are differentiated right (subscript R) and left (subscript L) based on functional rather than anatomical considerations. An anatomical representation would show crossed projections from each retina to contralateral tectum and both crossed and uncrossed projections from tectum to the motor area. The elements of the visual system of frogs and toads that have analogs in the model are the lens and its accommodation mechanism, the retina, the optic tectum, the nucleus isthmi, and the motor area.

Although there is only slight information available about the accommodation mechanism in frogs and toads [Grüsser & Grüsser-Cornehls, 1976], it is known that the two protractor lentis muscles move the lens along the optical axis to effect a change in focal length [Jordan et al., 1980]. Since adequate physiological data is lacking, we turn to the lens/prism studies of Collett [1977] for evidence that lens

Fig. 41 - A Possible Neural Realization of the Model.

This figure depicts our proposal for how the various functional components of the model could be distributed in the nervous system of the frog or toad. The imaging component is clearly located in the eyes. Pattern recognition and binocular prey selection are pictured as being carried-out in two layers of the tectum. Nucleus isthmi supplies both the binocular relay function and the global inhibitory function. Actual control of the lenses is pictured in the motor area. Labeling of the various components is identical to that in Fig. 30.


Eyes
Tectum
Nucleus Isthmi
Motor·Area
lenses
retinas
pattern recognizers
excitatory layers
binocular prey selectors
inhibitory layers
attention angle layers
accommodation control
R
L
R_R
T_R
B_R
N_R
U_R
R_L
T_L
B_L
N_L
U_L
θ_R
θ_L
Accommodation Control Signal

accommodation is sufficient to provide depth cues adequate for accurate prey snapping. In relating the lens accommodation system to the model we made the assumption that the lenses are coupled so that they both accommodate to the same depth. This assumption requires experimental verification.

There is indirect evidence to support the assumption that the strength of retinal ganglion-cell signals is a function of the degree to which the retinal image is correctly focused. The frog retina has been shown to contain at least four types of pattern selective units which project to tectum [Lettvin et al., 1959]. The receptive fields of retinal type-2 units, that provide the bulk of retinal input to tectum [an der Heiden & Roth, 1983], are small (4 to 8 degrees) and consist of an excitatory center with an inhibitory surround. This kind of unit should be maximally sensitive to small sharply-focused visual stimuli.

Each tectal hemisphere receives retinotopically organized stimulation from the contralateral eye. The construction and location of the eyes is such that there is a significant region of binocular overlap in the frontal superior visual field, and both tectal hemispheres receive retinal information from this region [Fite & Scalia, 1976; Grobstein et al., 1980].

The function of pattern recognition within tectum is well known and has been the subject of extensive study by ethologists [Ewert, 1976] and modelers [Ewert & von Seelen, 1974; Lara, Arbib & Cromarty, 1982; an der Heiden & Roth, 1983; Cervantes, Lara & Arbib, 1983]. Tectal units have been identified that are sensitive to object elongation along the axis of movement, direction of movement, figure-ground contrast, etc. In our model we assume that these feature detectors are able to isolate prey-stimuli from the visual background.

The source of the binocular information necessary to enforce agreement between the brain sides could be the nucleus isthmi. Each nucleus isthmi receives its input solely from the ipsilateral tectal lobe [Gruberg & Lettvin, 1980] and sends efferent projections to both the ipsilateral and contralateral tectal lobes. The relationship between tectum and nucleus isthmi is shown in block diagram form in Fig. 42. The tecto-isthmal and the isthmio-tectal projections are topographically organized so that the ipsilateral tecto-isthmio-tectal pathway maps a locus on tectum back to itself. The crossed connections map the binocular region of one tectal lobe to the binocular region of the opposite tectal lobe. These maps are in register (with respect to a fixed horopter surface [Gaillard & Galand, 1980]), and provide each tectal lobe with a source of binocular input. Crossed connections also exist between the monocular regions of the two tectal lobes. These map anatomically and not visuotopically similar loci to each other [Grobstein & Comer, 1983]. Lesions to either the nucleus isthmi or its crossed efferent pathway abolish ipsilateral tectal stimulation [Glasser & Ingle, 1978; Grobstein et al., 1978]. Thus, it is well established that the nucleus isthmi is the primary way-station in an intertectal binocular relay.

We assume that tectal binocular units receive three distinct excitatory inputs. Two of these inputs come from the two sides of the tectum via relays in nucleus isthmi with a third input from pattern recognition cells coming entirely from within tectum. This assumption is not based only upon the carefully preserved topography in the tecto-isthmo-tectal connections. The fibers of the crossed and uncrossed

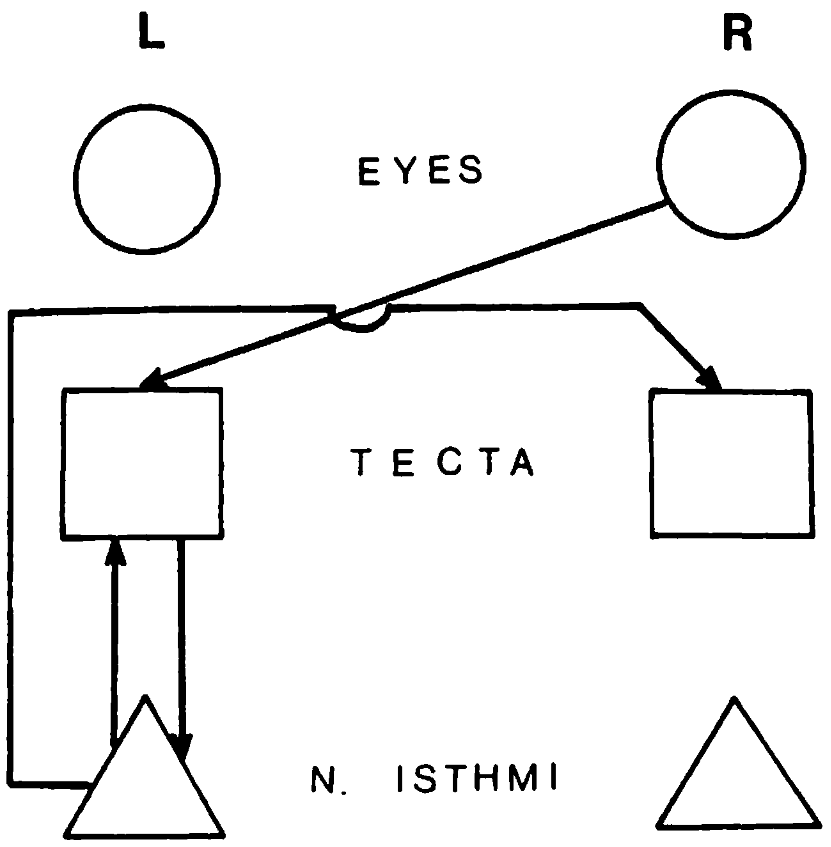


Fig. 42 - Block Diagram of Isthmio-Tectal Projections.

This diagram schematically shows the pathways from the right eye to the right (R) and left (L) tectal lobes. It illustrates both the ipsilateral and contralateral connections between the tectum and the nucleus isthmi. An identical set of connections exist for the left eye and right tectum. Reprinted by permission from Gruberg and Lettvin [1980].

isthmio-tectal pathways terminate in distinct tectal laminae, as shown in Fig. 43. Fibers from both nuclei terminate in the vicinity of layer 8 where there are few optic-nerve terminals. The fibers from the nucleus isthmi carry information originating only from tectal efferents and thus may be assumed to result from visual events "significant" to the animal. In particular, prey activity in the visual field would presumably excite a high degree of activity. Signals from both nuclei experience similar delays between the original optic input and their arrival in layer 8 [Udin, personal communication]. Thus, they are properly aligned both topographically and temporally, probably carry only visually significant information, and are free from the presence of direct optic-nerve input. This configuration seems particularly well suited for providing the binocular cross-coupling required by the model's prey selectors. We assume that the tectal binocular units receive this input across a broad receptive field [Fite, 1969; Raybourn, 1975].

Fig. 43 also shows that just below layer 8 in layer G is a region innervated by the optic-nerve but without input from the nucleus isthmi. This is what would be required to provide the input to the pattern recognizers, since to be effective they must have a fairly pure direct input from the retinal ganglion cells. We assume that tectal binocular cells receive input from tectal pattern recognition units across a narrow receptive field. We picture the binocular cells as responding optimally when they are receiving input from all three sources: ipsilateral nucleus isthmi, contralateral nucleus isthmi, and pattern recognizers. The inputs from nucleus isthmi weight cells in binocular correspondence, and the inputs from the pattern recognizers assure angular acuity.

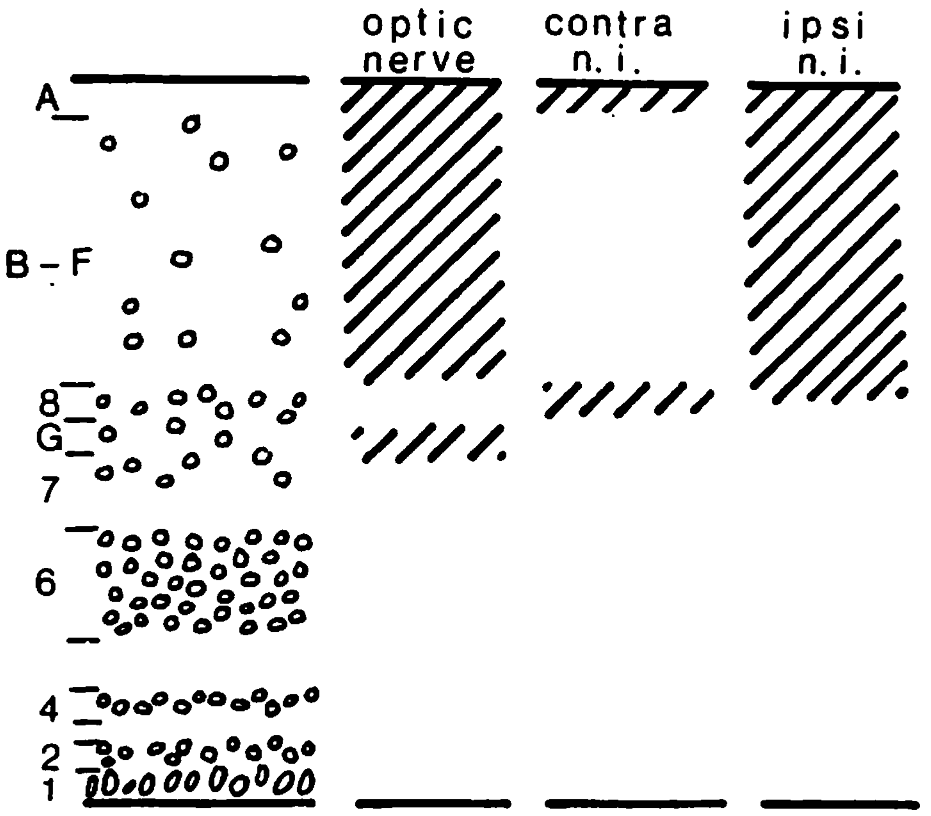


Fig. 43 - Distribution of Fiber Terminals in Superficial Tectum.

The layers of tectum are shown from the most superficial layer (A) to the ependymal layer (1). The shaded areas show the distribution of terminals of the optic nerve, the contralateral nucleus isthmi, and the ipsilateral nucleus isthmi. Of particular interest to the depth model are the layer labeled 8 where contralateral and ipsilateral isthmal input are present without optic-nerve input, and the layer labeled G where only optic nerve input is present. As explained in the text, layer 8 provides the right conditions for binocular cross-coupling, and layer G is well suited to provide input to the pattern recognizers. Reprinted by permission from Gruberg and Lettvin [1980].

Besides providing excitatory feedback to the tectal prey selectors, the nucleus isthmi may also provide the global inhibitory signal required by the prey selectors. There is some evidence that, as well as the isthmio-tectal projections providing the binocular relay, there is a second isthmio-tectal system providing tectal inhibition. Glasser and Ingle [1978] noticed that frogs with large lesions to nucleus isthmi show an elevated level of spontaneous tectal activity. Grobstein et al. [1978] report that there seem to be two distinct crossed projections from the nucleus isthmi. The projection involved in binocularity arises from the ventral and medial rim of the nucleus and projects to rostral tectum. A second projection arises from the medullary region of the nucleus and projects to the entire tectum. This projection may be inhibitory. For instance, although there is an anatomically well defined projection to monocular tectum, there have been no successful recordings made of activity in monocular tectum elicited by visual stimulation of the corresponding monocular region of the ipsilateral eye. Although this evidence is not conclusive, we make the assumption that the projection from the medullary area produces a global inhibition in tectum proportional to the total level of activity in the tectum. This assumption is also attractive from an anatomical point of view, since each of these nuclei take as their only input a topographic map of tectal activity, and concentrate this map into a relatively small region. Thus, they have the structure and neural pathways necessary to provide a signal to tectum that is modulated by the average (or total) firing rate across the entire tectal surface.

Finally, tectal efferents, pictured here as coming from broad-field binocular cells, descend into the motor area, where they initiate actions of the animal [Ingle, 1983; Grobstein et al., 1983]. We assume that one such motor activity is adjustment of the accommodative state of the lens and that this motor activity precedes any overt action in prey catching.

Suitability of the nucleus isthmi as a tecto-tectal relay for depth perception

If cross-tectal projections via the nuclei isthmi are to provide the binocular data for prey selection, they must be sufficiently dense to account for the accuracy in depth determination shown by both frogs and toads. Each nucleus isthmi contains about 8000 cells [Gruberg & Udin, 1978; Wang, 1981], and each of the crossed and uncrossed isthmio-tectal pathways contains about 4000 fibers [Gruberg, personal communication]. In order to estimate the error in angular disparity to be expected from this number of fibers, we make the following assumptions:

1) the fibers innervate the neural representation of a 180-degree (horizontal) by 90-degree (vertical) visuotopically organized section of a cone,[3]
2) the fibers are organized into evenly spaced concentric semicircles, each representing a fixed angular elevation in visual space, and
3) the number of fibers allocated to a particular elevation is proportional to the circumference of the semicircle representing that elevation.

[3] Gaillard and Galand [1980] describe the horopter in frogs as being cone-like.

A simple calculation, based upon these assumptions, shows that if the ground-plane semicircle is allocated 180 fibers (one per horizontal degree) then nearly 45 such rings can be constructed from 4000 fibers. Thus, the horizontal angular resolution on the ground-plane will be ±1/2-degrees. The vertical resolution will be ±1-degree. The worst-case horizontal disparity error on the ground plane, combining two such representations, will be twice the maximum error in horizontal acuity or ±1-degree. Fig. 44a shows the approximate relationship between angular disparity and depth for objects located in the center of the binocular field. This curve was calculated using simple pin-hole optics and an assumed eye separation of about 3 cm. Fig. 44b, computed from the slope of the curve in Fig. 44a, shows the expected maximum depth resolution error as a function of depth. From this curve we see that the effect on depth estimation of the cumulative 1 degree error in angular disparity would be less than 0.2 centimeters for objects at 5 centimeters from the animal. At 10 centimeters the depth error increases to 0.6 centimeters and at 15 centimeters it is just greater than 1 centimeter. These figures compare well with animal behavior. For instance, the prey overshoot observed by Collett [1977] was between 0 and 2 centimeters for prey distances out to 15 centimeters.

Evaluation of the model

Although the model can be directly related to neurophysiological and neuroanatomical data, the algorithm derived from the model is interesting in its own right. In particular, it may be useful as a mechanism for depth perception in robots when the problem is to detect and localize a particular object. Simulation results demonstrate that this algorithm is able to rapidly locate a single object placed

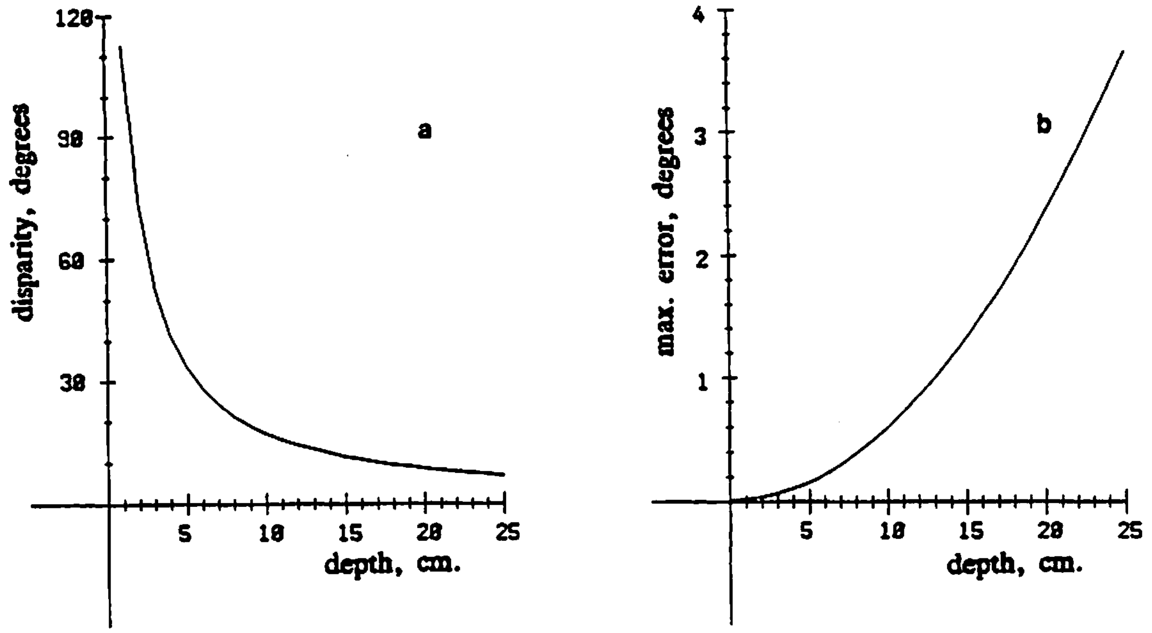


Fig. 44 - Angular Disparity vs. Distance from Viewer.

(a) Angular shift between the two retinas of the projection of a centrally-located point. This shift or disparity is displayed as a function of distance of the point. Distance is measured along the perpendicular drawn from the mid-point of the line connecting the two pupils. Assumptions of pin-hole optics, spherical retinas, and an eye separation of 3 cm. were used to compute this curve.

(b) Depth resolving error as a function of depth. This curve was developed on consideration of the number of fibers (4000) from the nucelus isthmi and assuming that these fibers innervate a 180 x 90 degree hemisphere.

---

anywhere in the binocular visual field, and select and localize the nearer of two identical objects when they are presented at different depths. Further, the algorithm is successful 75% of the time in selecting and localizing one of two objects over a large range of perfectly symmetrical presentations.

In toads, Collett noted a 94 percent effect from binocular cues and a 6 percent contribution from monocular cues. However, when single prey presentations are made to our model, lenses have no effect upon depth resolution whereas prisms cause the expected shift in depth estimation. This is because the model contains no mechanism to allow for any direct monocular contribution to depth estimation. It computes its depth-estimate solely from binocular parallax. Thus, an obvious extension of the model would be to provide a secondary depth estimator from image focus cues. Since frogs and toads are successful at monocular prey-localization it is likely that such a system exists. However, our model demonstrates that optimizing image focus is not the only possible mechanism for controlling lens accommodation. In fact, it is likely that in frogs and toads both image focus and binocularity play a cooperative role in adjusting the lens.

Our experiments with simulated concave lenses and symmetrically presented prey showed that the model is prone to making both "crossed-ghost" and "binocular-average" errors under these conditions. On the other hand, toads seem to make only "binocular-average" errors under identical circumstances [Collett & Udin, 1983]. If the model employed both binocular and monocular cues to adjust the lens and estimate depth then the number of "crossed-ghost" errors would diminish and the frequency of "average" errors would increase. In other words the depth estimate from binocular mismatch would be offset by the depth estimate from lens focusing. Toads rapidly learn to avoid snapping altogether under these conditions [Collett & Udin, 1983]. This fact also argues for a dual-cued system. The discrepancy between the depth estimates provided by these two cue sources would be sufficient to "abort" the toad's normal prey-catching activity.

Of the categories of depth estimate given in Fig. 33 both the "uncrossed-ghost" and "ipsilateral-average" categories can provide depth estimates that are behind the eyes. A more sophisticated accommodation controller could be designed to improve performance based on this fact. Such a controller would use the depth setpoint, calculated from disparity in the prey selectors, only if its value were reasonable.

CHAPTER V

TOWARDS A COMPLETE MODEL OF DEPTH PERCEPTION IN FROGS AND TOADS

Summary

In this chapter we compare and contrast our two models, first with each other and then with data from biological experiments. Our analysis suggests that frogs and toads do not use a single means for determining depth but, instead, use two different processes, one for determining depth of stationary environmental objects and the other for determining depth of prey. We present an accommodation control system that meets the needs of both of these processes, and show how some of the discrepancies between animal experiments and our "prey localization model" would be eliminated by the nonlinearities inherent in this control. The chapter concludes with an evaluation of the depth scale used in the two models.

Introduction

In Chapters 3 and 4 we developed two quite different models of depth perception. The "cue interaction model" of Chapter 3 uses depth cues from both binocular disparity and lens accommodation to build maps that give the spatial locations of all objects in the visual field. Its computational process is similar to that of other cooperative depth models [Dev, 1975; Marr & T. Poggio, 1976; Amari & Arbib, 1977] but differs in that it uses two sources of depth cues. The "prey localization model" of Chapter 4 locates the point-in-space occupied by a single selected prey object. It does this by utilizing two binocularly cross-coupled prey selectors, one for each eye, in a feedback loop with the lens accommodation controller. The prey selectors provide the position of the selected prey on the left and right retinas. The angular disparity between these two positions is a measure of depth, and is used as a setpoint for accommodation. Accommodation, in turn, helps to assure that the two prey selectors agree on the same visual object.

In this chapter we attempt to complete our picture of depth perception in frogs and toads. To do this we compare the two models and then present the beginnings of a new model that is both a synthesis and an extension of the earlier models.

The Cue Interaction and Prey Localization Models

Unities

The "cue interaction model" shares several features with the "prey localization model." These commonalities are not accidental but, rather, point to certain underlying charactersitics of the depth resolution problem faced by frogs and toads.

The most important and most obvious similarity is that both models use lens accommodation to help disambiguate the binocular matching process. Further, it is depth from binocularity that determines the final depth output from each of the models. Obtaining depth from binocular matching is ambiguous without auxiliary depth cues, and such cues are available from lens accommodation in both frogs and toads. The models demonstrate two different ways in which image focus can be used to disambiguate the cues from binocular matching. The depth maps computed by the "cue interaction model" are dominated by binocular cues, and in the "prey localization model" the final lens accommodative state is determined by the binocular disparity between selected points. That the final depth estimate should come from binocularity and not lens accommodation is based upon both experimental findings [Collett, 1978] and theoretical considerations [Collett & Harkness, 1982].

The models are also similar in their computational schemes. They are both based upon competition and cooperation. Arbib [1981] makes a convincing argument that cooperative computation is the style of processing in the brain. Amari and Arbib [1977] provide a short review of some cooperative models and show how these models can be unified by a consistent mathematical formalism. Their presentation is particularly applicable to the problem of depth perception. Marr and T. Poggio

[1979], on the other hand, argue that cooperative computation is not necessarily the style of computation used for the depth perception (for a counter argument see Frisby and Mayhew [1980]). However, since toads can orient to the position of a prey even after it is no longer visible [Collett, 1982], whatever computational style *is* used must provide some form of short-term memory. Due to their inherent hysteresis, cooperative-competitive schemes naturally provide a stable interpretation of sensory input.

A more subtle similarity between the models is that while both models depend on binocularity for determining depth they achieve angular acuity by sampling the visual input at a point before any binocular interaction takes place. In the "cue interaction model" the retinal position of a point of visual input is used to index a depth map to determine the spatial position of the point. Retinal position is required since the depth maps produced by the model have good depth acuity but poor angular acuity. In the "prey localization model" monocularly driven pattern recognizers provide a precise angular position signal to the prey selectors. The selectors' binocular inputs are more broadly spread in the angular direction. These precise angular signals, which both models use, do more than provide directional acuity. Their source is close to the retinal input so they are relatively free from hysteresis. Thus, although neither model was designed to address the problem of changing visual stimuli, they both contain a mechanism that would allow them to be responsive to change.

Diversities

Although the two depth models have several commonalities they are more easily contrasted than compared. All of the important differences between the models stem from the fact that depth perception is viewed as an information-gathering process in the "cue interaction model," whereas in the "prey localization model" it is represented as an action-oriented information-seeking process.

In the "cue interaction model" there is no implication that any action is taken based upon the data present in the visual input. This model represents the information flow in the nervous system as being unidirectional. Any action based upon the output of the model is presumed to be taken by motor centers downstream of the depth mapping process. If, for example, a new object were introduced into the visual field the response of the "cue interaction model" would be merely to adjust its depth map.

The "prey localization model," on the other hand, utilizes an active coupling between sensory and motor processes. It represents the levels of the visual system as being part of an information-seeking feedback loop. Information flow is circular, completing a cycle of action and perception that is specialized to the capture of prey. In this model, a prey object introduced into the visual field would either be ignored, if the new input were less salient than a currently selected one, or would itself become the selected object and cause an adjustment of the accommodative state of the lenses. This adjustment would serve both to verify the binocular depth estimate of the object and to improve its visual acuity. Thus, rather than passively gathering data, this model pictures the visual system as commiting its resources to the examination of an object.

Since the "cue interaction model" is structured around the production of depth maps and the "prey localization model" is designed only to localize a single point, the two models require very different lens accommodation mechanisms. The "cue interaction model" requires an accommodation system that is able to continuously provide a depth likelihood (or image-focus) measure at every retinal position, and for all states of lens accommodation. An accommodation controller meeting this requirement would have to combine some form of accommodation scanning mechanism with a short-term memory, or map, recording image intensity at each lens setting. On the other hand, the accommodation system used by the "prey localization model" simply computes a single depth from binocular disparity and then adjusts the lens to bring that depth into clear focus.

A synthesis

Even though the two models represent very different ways in which both binocularity and lens accommodation can be used to determine depth, it is not necessary that we choose one of these mechanisms over the other. Instead, we will argue that the differences between these models specialize them for two equally important roles, and that mechanisms similar to those used in both models are essential to support the full range of activities exhibited by frogs and toads. Our conclusion is that multiple depth-resolving systems may coexist in a single animal.

To successfully navigate through their environment frogs and toads would need a reasonably complete "picture" of the spatial configuration of surfaces in that environment. Toads are able to simultaneously perceive the positions of several barriers, gaps, and chasms. During prey approach behavior they coordinate their activity based upon rules that take into account the complete spatial configuration confronting them [Collett, 1982; Arbib & House, 1983; Lara et al., 1983]. The "cue interaction model" builds the kind of spatial depth map needed to support this observed behavior. It is also best suited to the kind of visual input presented by natural barriers. This input, from grasses, twigs, rocks, etc., would more closely resemble a random-dot stereogram than would the very sparse input from prey. The algorithm underlying the "cue interaction model" [Dev, 1975; Marr & T. Poggio, 1976; Amari & Arbib, 1977] was originally designed to explain perception of depth in random dot stereograms and is especially effective with this kind of data. Since the "prey localization model" locates only a single point it is ill-suited for determining the depth of barriers.

For catching prey the advantage lies not in determining the positions of all prey-like objects but in chosing a single prey and then directing all sensory and motor activity towards its capture. The "prey localization model" is best suited for this form of depth resolution. It selects a single prey and then maximizes the quality of the image of that prey by adjusting the lens focus. It even has a good source for a cue to initiate a strike at the prey; when the lens accommodative state matches the depth estimate from binocularity there is a high degree of certainty that a correct depth measurement has been reached and a strike should be initiated. The more complex depth-mapping machinery of the "cue interaction model" is neither necessary nor efficient for prey capture.

Our hypothesis, that prey and barrier depth are determined by two different processes, is also supported by anatomical, physiological, and behavioral evidence.

A large number of experiments indicate that prey acquisition in frogs and toads is tectally mediated, whereas barrier avoidance is mediated by the thalamus. Extensive studies have been made of tectal units sensitive to small moving prey-like objects [Ewert, 1976; Ewert, 1982]. Functional units specialized for stationary edge detection have been discovered in thalamus [Ewert, 1971, Brown & Marks, 1977; Ingle, 1980]. The blue-sensitive units discovered by Muntz [1962] also respond to stationary input. Frogs with bilateral tectal ablations do not exhibit prey catching behavior but do retain the ability to avoid barriers [Ingle, 1977]. Motor pathways reflect the same organization. Severing the tecto-fugal pathways disrupts orientation and snapping behavior, whereas severing the pretecto-fugal pathways disrupts sidestepping to round barriers [Grobstein et al., 1983; Ingle, 1983]. There is partial evidence that this differentiation of function extends to depth perception; atectal frogs are able to discriminate the depth of barrier surfaces [Ingle, 1982].

The functional differentation between tectum and thalamus is matched by anatomical differences that also support our hypothesis. The computation done by the "cue interaction model" is most easily supported by a structure, like the thalamus, that has overlapping representations of the visual fields of both eyes [Scalia & Fite, 1974]. The "prey localization model" is best suited to the organization exhibited by the tectum and nucleus isthmi, in which the tectum receives only monocular projections but the nucleus isthmi acts as a relay to pass information between tectal lobes [Glasser & Ingle, 1978].

Our proposal for how the depth mapping and the prey localization processes might be distributed in the brains of frogs and toads is shown in Fig. 45. The thalamus is shown receiving a binocular projection from the eyes as well as signals indicating the lens settings being called for by the accommodation controller.[1] These inputs would provide the thalamus with the information required to make depth inferences based on both binocularity and lens accommodation. A process similar to that used in the "cue interaction model" could use these inferences to build depth maps. The tectum is pictured as receiving monocular projections, but also has relay pathways through the nucleus isthmi that allow information interchange between the two tectal lobes. As in the "prey localization model," projections from tectal prey selectors would provide the accommodation controller with information for computing a binocularly-derived depth setpoint.

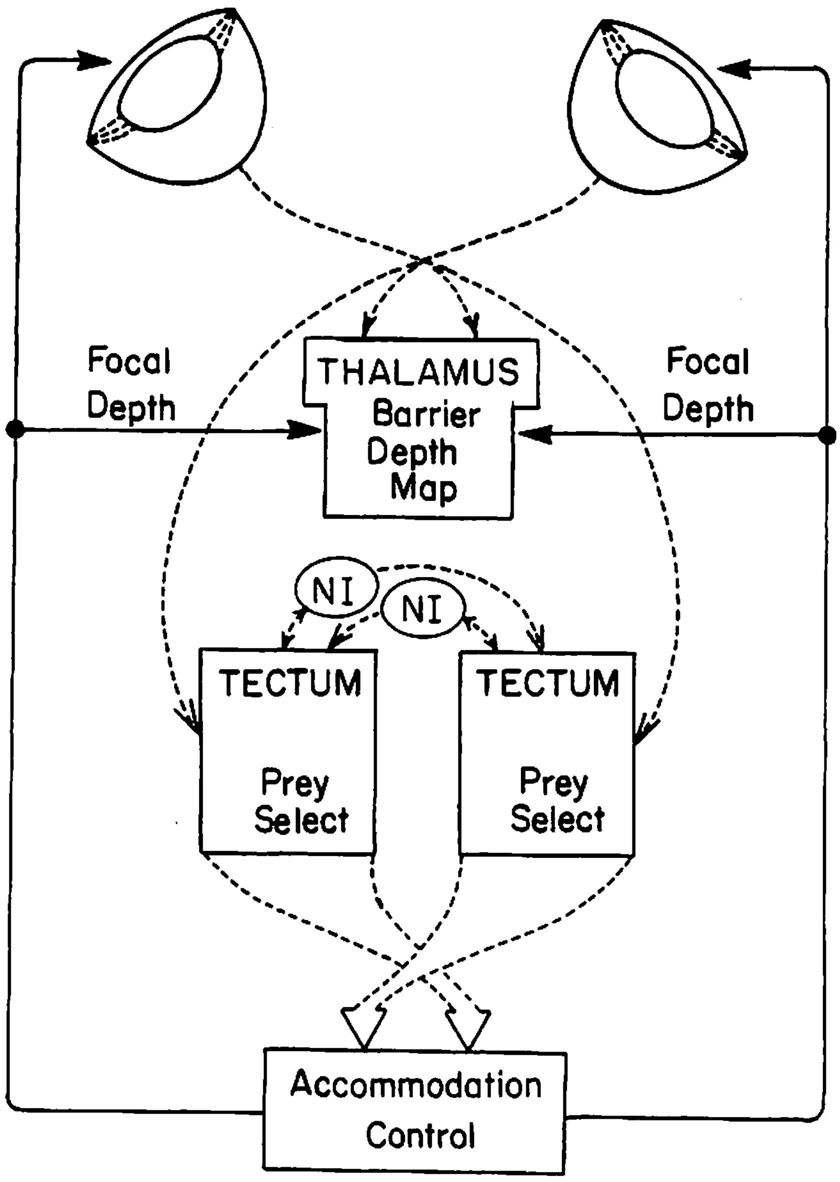
An Extended Model of Accommodation Control

The hypothesis that these two different depth perception processes coexist in frogs and toads requires that we consider a more sophisticated model of accommodation control than was posited in either of the two previous models. If we assume that accommodation is used to provide monocular depth cues that assist in determining the depth of both barrier and prey objects then the accommodation control system must be capable of at least two modes of operation.

[1] Rubinson and Colman [1972] report that the nucleus posterolateralis in the thalamus of frogs (*Rana pipiens*) receives an afferent projection from the midbrain tegmentum. However, its physiological function is unknown.

Fig. 45 - A Possible Anatomical Distribution of the Two Depth Perception Processes.

This diagram shows how both depth mapping of barrier surfaces and localization of prey may be distributed in the nervous system of frogs and toads. Eyes are represented at the top of the figure, and internal brain regions are indicated by labeled rectangles. Dashed arrows depict information inflow, and solid lines represent outflow. The thalamus receives overlapping projections from both eyes as well as signals indicating the current lens accommodative state. These inputs would allow the development of a depth map of visual space using a mechanism similar to that outlined in the "cue interaction model." Each tectal lobe receives only a monocular projection from the contralateral eye, but ipsilateral information is available from tecto-tectal relays through the nucleus isthmi (NI). These inputs provide tectal prey selectors with the information necessary to localize a single prey. Prey selector output projects to the accommodation controller.


Focal Depth
THALAMUS
Barrier Depth Map
Focal Depth
NI
NI
TECTUM
Prey Select
TECTUM
Prey Select
Accommodation Control

The "cue interaction model" requires that the brain be able to construct a map of image intensity indexed by both retinal position and lens accommodative state. In order to construct this type of map the lens would have to be scanned, so that over a period of time a complete map would be developed. This mode of accommodative control is appropriate for the depth mapping of stationary barrier-like objects, but is unlikely to be effective for the mapping of moving prey-like stimuli.

The "prey localization model" requires that the accommodation controller be able to quickly lock-on to a prey stimulus, setting the lens accommodative state to match the true depth-in-space of the prey. This mode of accommodative control is appropriate for isolated stimuli, and its speed would allow it to be responsive to a moving stimulus.

Even if the accommodation controller were to have the two modes of operation described above it would still not be capable of representing some of the more subtle depth-related behaviors exhibited by frogs and toads. Toads judge the depth of prey located in their frontal binocular field using mostly binocular depth cues. Nevertheless, their depth estimates do include a small (6%) effect from accommodation. When concave lenses are placed in front of their eyes or when drugs are used to disrupt accommodation, binocular toads make small errors in depth estimation [Collett, 1977; Jordan et al. 1980]. The "prey localization model" does not replicate these errors since it does not use monocular cues in its estimation of depth. The question remains: how are accommodation and binocularity integrated for prey catching?

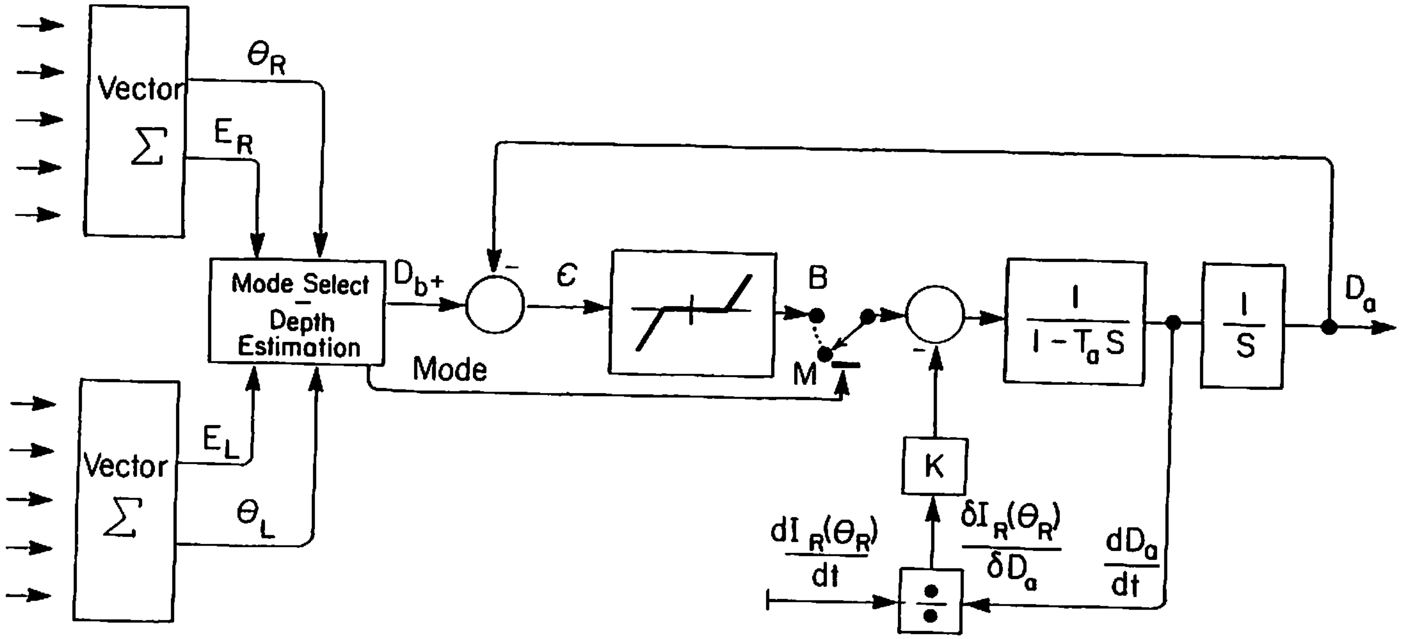
In order to address this question we present the following findings from experiments by Collett and Udin [1983]. Over a range of experimental conditions, estimates of the depth of a single prey made by toads with lesions to the nucleus isthmi are very similar to those made by intact toads. But there are important differences between the estimates made by lesioned and unlesioned animals when two prey are presented simultaneously. In this case, the lesioned toads will sometimes snap at a "ghost" prey located between the two true prey, and near to the point-in-space corresponding to the position predicted from binocular mismatch (see the "crossed-ghost" error of Fig. 33b). However, the error they make is not *exactly* that predicted by binocular mismatch; snaps made toward "ghost" prey consistently overshoot the depth expected from a true binocular mismatch error. When concave lenses are mounted in front of their eyes the lesioned toads see "ghosts" at the same distance as they do without the lenses. However, when the lesioned toads view two prey through lenses and select a *real* prey, their snaps fall short of the true target but are considerably further than their snaps at "ghosts." This result is seen only in nucleus-isthmi lesioned animals, and is similar to results obtained in earlier experiments with monocular toads [Collett, 1977].

Our "prey localization model" predicts "ghost" snapping when the nucleus isthmi is lesioned but it does not predict the more subtle effects apparently due to the interaction between binocular and accommodation cues. To realize these effects we propose three additional extensions to the accommodation controller. First, we introduce a deadband into the controller so that control action will be highly responsive to large errors between the binocular setpoint and the current lens setting but will be unresponsive to this error when it is small. Second, we introduce a control signal derived from image focus cues that will act to improve the focus of the retinal image. Third, the use of image focus cues requires that the dynamics of the controller be replicated so that there is a separate control path for each lens. If only binocular cues are used by the controller both lenses will receive the same depth setpoint, so there need be only one control path. However, image focus cues will be different for each eye, requiring two separate control paths. With these additions in place, the controller could be adjusted to use binocularity to achieve a rapid "approximate" lens adjustment and focus cues to make a further "fine" adjustment.

The fully extended version of the accommodation controller is shown in Fig. 46. As in the accommodation controller of the "prey localization model," the vector summation boxes receive input from the tectal prey selectors. In addition to providing retinal positions θ_L and θ_R, denoting the centers of gravity of excitation in the two prey selectors, the summers also provide level signals E_L and E_R, that measure the intensity of excitation in the prey selectors. The "mode-select" box has two outputs: D_b, the binocular depth setpoint computed from the difference (or disparity) between positions θ_L and θ_R; and a mode selection signal, that switches the controller between monocular (M), and binocular (B) modes. The error signal ϵ, the difference between the depth setpoint D_b and the current accommodative setting D_a, is shown passing through a deadband that effectively decouples binocular control when lens accommodative state and the binocular setpoint are nearly identical. The controller is also augmented by a circuit that measures the change in image intensity

Fig. 46 - Extended Accommodation Controller.

This version of the accommodation controller extends the controller used for the "prey localization model" in four ways. First, a mode selector and switch toggle the controller between binocular (B) and monocular (M) modes. With the switch in position B the controller is driven by the error signal ϵ between the binocular setpoint D_b and the current controller output D_a. With the switch in position M the binocular setpoint no longer has an effect on the controller. Second, a deadband is placed in the error signal path so that binocular control is also decoupled when the error is small. Third, a signal proportional to the change in image intensity I with respect to change in controller output D_a provides a feedback that acts to improve image focus in a "hill-climbing" fashion. If the gain K of this focus feedback is made small enough the controller will be dominated by the binocular error signal ϵ as long as the controller is in binocular mode (B) and the error signal is outside the deadband. However, if the controller is in monocular mode (M) or or the error signal is within the deadband then the focus cues will determine the controller's output. In order to avoid clutter, the fourth extension to the controller is not shown on the diagram. This extension is simply to replicate the controller's dynamics so that there are independent control paths for each eye. In binocular mode this makes little difference, since both sides would receive the same binocular setpoint D_b and would, therefore, be driven to the same depth, modulo any small differences due to image focus cues. However, in monocular mode the image focus cues would be derived from each eye separately, so that the accommodative states of the two lenses could be quite different.


Vector
Σ
θ_R
E_R
Mode Select Depth Estimation
D_b +
−
ϵ
B
M
Mode
$\frac{1}{1-T_a S}$
$\frac{1}{S}$
D_a
Vector
Σ
E_L
θ_L
K
$\frac{dI_R(\theta_R)}{dt}$
$\frac{\delta I_R(\theta_R)}{\delta D_a}$
$\frac{dD_a}{dt}$
÷

with respect to change in lens accommodative state $\frac{\delta I_R(\theta_R)}{\delta D_a}$. This circuit provides positive feedback through a gain K, so that when the system is decoupled from binocular control (either because of the mode switch or the deadband) the lens will be adjusted in a "hill climbing" fashion to improve image focus.

There are various ways in which mode selection could be done in the controller. The simplest would be to compare excitation levels E_L and E_R with small thresholds, with monocular mode selected if either level is below threshold and binocular mode selected if both are above threshold. In addition to thresholding, a more sophisticated mode selector could apply consistency constraints. For instance, a requirement could be that both excitation levels have similar values. Another could be that the depth setpoint D_b computed from the difference between positions θ_L and θ_R correspond with a location in the frontal binocular field of the animal.

In the "prey localization model" it is assumed that the two lenses are coupled so that they are always adjusted to the same accommodative state. In this extended version of the controller this constraint is relaxed. The dynamics of the controller are replicated, so that there is one control path for each lens. When accommodation is under binocular control both paths receive the same setpoint D_b so that their action is coupled. When under monocular or image focusing control each control path acts to independently maximize the clarity of focus of the image on the side which it is controlling.

In order to support its monocular focusing mode the controller would require two very different kinds of input from higher brain centers. First, it would require inputs that are not heavily processed, and, therefore, closely mirror the changing pattern of retinal excitation. These inputs would provide the controller with the sensitivity to retinal activity required to detect small changes in the quality of image focus. Projections to the controller directly from thalamic and tectal pattern sensitive cells could provide this sort of input. The controller would also require inputs that would govern which local regions on the retinas should be monitored for determining the quality of focus. The tectal prey selectors, already included in the accommodation controller of the "prey localization model," could provide this input for prey stimuli. For barrier stimuli, signals indicating the most salient barrier edges would be sufficient.

In frogs and toads there are tectal and thalamic efferent pathways that could supply these two kinds of input. Tectum and thalamus each have two major efferent pathways to the brain stem. In both cases, one of these pathways has its origin close to the terminal arborizations of the optic nerve, and the other pathway has its origin further from these terminals [Ingle, 1983; Grobstein et al., 1983] (see Fig. 9 in Chapter 2).

The augmented accommodation controller, described above, has a structure that would allow it to replicate the data from behavioral and lesion experiments. First, the controller could be tuned to have a response that is dominated by binocular cues when they are available, but would also provide for a small contribution from monocular cues. If the gain K on the image focus feedback is kept small and the controller is in binocular mode then this feedback will have only a negligible effect as long as the error signal is large enough to be outside of the deadband. But when the error is reduced so that it is within the deadband the focus signal will have its full effect. However, the range of this effect is limited by the deadband. If the focus mechanism adjusts accommodation enough to move the error outside the deadband binocular control will be reinstated and return the error to within the deadband. If we assume that the depth estimate used by the animal for snapping is the controller's output signal D_a, then the presence of concave lenses in front of the eyes would cause undershooting, but only to the extent permitted by the width of the deadband. If the "crossed-ghost" error were being made by the binocular system, as in nucleus-isthmi lesioned toads, then the focusing system would tend to move the lens to correct this error but, again, this adjustment could only go as far as the deadband and binocular control would allow. The resulting depth estimate would be somewhat farther away than the estimate that would have been made by binocularity alone. Concave lenses placed in front of the eyes would have no effect on the estimation of the depth of "ghosts," since the misfocus due to binocular error would be much greater than any effect from the lenses. The effect of lenses on the estimation of the depth of real (as opposed to "ghost") targets in lesioned toads could be explained by the separation of the accommodation control paths for the two eyes. If the strength of the concave lenses were great enough, then the defocusing effect might be enough so that one side of the visual system, presumably the side ipsilateral to the selected prey, would have the object in better focus than the other side. The difference in strength between the signals from the two prey selectors could be interpreted by the controller as a binocular mismatch. The binocular decoupling due to the lesions to the nucleus isthmi would reduce the strength of the input to the prey selectors increasing the likelihood of this mismatch.

Because of this mismatch, the controller would revert to monocular mode and the concave lenses would have their full depth distorting effect, as if the animal were monocular.

We complete our description of the accommodation process by examining the effect that the addition of focus control would have when the controller is being driven in the accommodative scanning mode. In this mode, the presence of image focusing feedback would cause the lens to spend longer periods of time in the regions of its cycle where image intensity cues are greatest. Instead of passively scanning, the controller would be more highly responsive to regions of the image as they came into clear focus.

Experimental Verification of the Models' Depth Scale

There is experimental evidence providing a confirmation of the internal depth scale used in both of our models. The models use a disparity-based scale (see Appendix B), rather than a Cartesian scale, not only to represent depth estimates obtained from binocular disparity but also to represent depth estimates from lens accommodation. One result of using this scale is that depth estimates become progressively less acute with distance from the eyes. That there should be a reduction of depth resolution with increasing distance is intuitive. However, does a disparity-based depth measure provide the correct form for this nonlinearity? Jordan et al. [1980] used Miotic, a muscle contractor, and Atropine, a muscle relaxor, to disturb the action of the accommodation muscles in toads (*Bufo bufo*). They compared the maximal snapping distance[2] (MSD) of treated animals with that of untreated animals. The average MSD for untreated binocular toads was 4.3 cm. When treated with Atropine the average MSD increased to 5.0 cm., and when treated with Miotic it decreased to 3.7 cm. In monocular toads the difference was more extreme. Their average MSD when untreated was 4.6 cm, compared with 7.2 cm. when treated with Atropine and 3.2 cm. when treated with Miotic. Table 1 summarizes these data, and how they would translate onto a disparity scale. If disparity is taken as the depth measure, the percentage contributions of Miotic and Atropine to the full drug-induced change in depth estimate (Atropine depth – Miotic depth) are nearly the same for both binocular and monocular toads. In both cases Atropine contributes about 47% of the change and Miotic contributes about 53%.

Table 1 also suggests reasonable upper and lower limits for the dead-band that is applied to the error signal of the accommodation controller of Fig. 46. When they are subjected to full disruption of accommodation, binocular toads make depth estimation errors which are equivalent to a 2.8° error in retinal angle. Thus, we may assume that binocularity will bring the lens to within ±2.8°, after which the final lens position, and depth estimate, is based upon maximizing image focus.

[2] Maximal snapping distance is the distance at which toads first snap at prey when the prey is moved slowly towards them. If a perturbation of the visual system increases an animal's MSD then it will tend to undershoot prey (it snaps when the prey is farther than it "thinks"). If the MSD is decreased the animal overshoots.

Table 1 - MSD After Drug Treatment of Accommodation Muscles.

BINOCULAR TOADS

	Untreated	*Atropine*	*Miotic*
MSD, cm.	4.3	5.0	3.7
ΔMSD, cm.	---	0.7	−0.6
% full range (1.3, cm.)	---	54	46
Θ, $\tan^{-1}$(MSD / 1.5)	70.7°	73.3°	67.9°
ΔΘ	---	2.6°	−2.8°
ΔD, 2ΔΘ / 90°	---	0.058	−0.062
% full range (0.120)	---	48	52

MONOCULAR TOADS

	Untreated	*Atropine*	*Miotic*
MSD, cm.	4.6	7.2	3.2
ΔMSD, cm.	---	2.6	−1.4
% full range (4.0, cm.)	---	65	35
Θ, $\tan^{-1}$(MSD / 1.5)	71.9°	78.2°	64.9°
ΔΘ	---	6.3°	−7.0°
ΔD, 2ΔΘ / 90°	---	0.140	−0.156
% full range (0.296)	---	47	53

Distance data from Jordan et al. [1980]

Calculations of angles were made assuming an interpupillary distance of 3.0 cm, and MSD measured along the midline. ΔD is change in disparity between the retinal projections of a point on the right and left retinas, with a change of 90° represented by a 1 unit change in disparity.

Discussion

We have argued that frogs and toads probably have at least two kinds of depth perception, one for deriving a depth map of stationary environmental objects, and the other appropriate for localizing and snapping at moving prey. If this dichotomy does, in fact, exist then it is likely that the mapping process takes place in the thalamic region of the brain and the prey localization process takes place in the tectum.

Although we have suggested only two depth perception systems there is no reason to suppose that our arguments could not be used to support even more systems. Along with certain anatomical, physiological, and behavioral evidence, our argument rests mainly on the grounds that perception is action oriented. Different actions require different depth information. Thus, what is necessary and efficient for one type of action may be either unnecessary or inefficient for another type of action. This argument may, of course, be extended to cover behaviors other than prey catching and barrier negotiation, such as mating, locating ponds, and journeying. The potential for an explosion in the number of depth perception processes required by a single animal suggests that theorists should attempt to determine a minimal set of such processes that could efficiently support the wide variety of behaviors requiring depth perception. We have suggested two such processes 1) derivation of a complete environmental depth map, and 2) localization of a single point.

We have constructed a preliminary model for a lens accommodation controller that would be able to support both of these processes. This model contains a method for switching between a binocularly driven "prey selection" mode and a "barrier mapping" mode driven by image focus cues. The model is also configured to explain some of the effects seen in behavioral experiments with toads, when the correspondence between monocular and binocular depth cues is disrupted by placing concave lenses in front of the animal's eyes.

Behavioral data support the idea of a continuously developed depth map in thalamus. Atectal frogs are able to discriminate the depth of barrier surfaces during escape [Ingle, 1982] (see Fig. 10). Escape requires that an animal be able to respond immediately, and not after patiently constructing a depth map.

However, our assumption, that a depth map of stationary environmental objects is continually being built, seems to contradict physiological evidence that frogs and toads show very little visually elicited neural activity when there is no motion in the visual field. Our reply to this seeming contradiction has two parts: 1) we claim that this depth mapping process goes on in the thalamus, where at least two kinds of neural units responsive to stationary visual input have been found; and 2) experiments indicating that motion in the visual field is required to elicit neural activity have been done with immobilized animals and are far from conclusive.

The thalamic units responsive to stationary objects are the blue sensitive *on* units first reported by Muntz [1962], and the stationary edge detectors best described by Brown and Marks [1977]. Of these, the blue sensitive *on* units appear to be clustured in the region of the nucleus of Bellonci [Brown & Marks, 1977], which is one of several binocularly innervated sites in the thalamus [Scalia & Fite, 1974].

Most electrophysiological preparations encountered in the literature involve the use of curare to paralyze the animal. But curare is known to have direct effects on such physiological parameters as excitatory receptive field separation between ipsilaterally and contralaterally activated tectal units [Raybourn, 1975], and receptive field sizes of units in the nucleus isthmi [Gruberg, personal communication]. Besides these direct effects, the use of curare is bound to inhibit lens accommodation, which could, by itself, induce changes in the light pattern falling on the retina and elicit neural activity.

Besides using drugs to paralyze or quiet experimental animals, most electrophysiological experiments are done with the animals firmly mounted to an apparatus, so that no head or body motion is possible. It is well known that freely functioning frogs and toads, even when in a stationary pose, make eye movements that are synchronized with their breathing [Schipperheyn, 1963]. These mounting techniques effectively eliminate any motion of the eyes induced by body movements.

In short, experimental techniques are not adequate to allow the inference that freely-functioning frogs and toads are essentially blind unless there is motion in the visual field. It is our opinion that in visual animals, like frogs and toads, the visuo-motor system is highly integrated, so that disrupting motor function will strongly influence visual function.

CHAPTER VI
CONCLUSIONS

In our introductory chapter we expressed the hope that by modeling a natural system we could make contributions to the fields of both brain theory and robotics. Specifically, we wished to contribute to the understanding of the depth perception process in frogs and toads and, at the same time, to suggest depth algorithms that would be useful in the design of robotic systems. But our initial hope must be tempered by the fact that the disciplines under consideration are both in early and exploratory stages. Except under highly constrained conditions robots do not yet make successful use of sensory feedback to control their actions. Likewise, attempts to relate animal behavior to underlying neurophysiology and neuroanatomy entail making assumptions that are often untestable using current methods. It is clear that at this stage of development a study such as ours can make only preliminary claims.

Some would argue, then, that work of this nature is premature, that successful modeling must await major advances, particularly in the neurosciences. Here we disagree strongly. The brain is so complex and experimentation is so difficult that progress in the neurosciences cannot be made without guidance from theory. Theory, on the other hand, is often far too abstract to provide the kind of concrete hypotheses that can be readily subjected to experimental testing. Modeling, particularly when it is accompanied by simulation, is a process of transforming abstract theory into testable structures. To the extent that the structure and performance of a model conform to current theory and available experimental data it acts as a confirmation of both the theory and the data. Where structural constraints from experimental data are lacking the modeler must make assumptions that, by themselves, are a rich source of hypotheses for experimental testing. Further, simulation results provide predictions about the expected performance of the modeled system. This is especially important when performance data are lacking or difficult to obtain. In turn, as better theory is developed or new experimental data becomes available a model must be updated, refined or replaced. We feel that the attainment of lasting and solid results depends upon a continuing cycle of theory, modeling, and experiment; biological investigation, and machine simulation. To slight any one of these avenues must seriously hamper activity in the other areas. The proper role for modeling, especially at this stage, is to provide enough of a synthesis of the information from the other disciplines to allow the formation of hypotheses that are capable of guiding further experimentation.

This final chapter reflects the role that we envision for modeling. It very briefly reviews our models but, more importantly, it outlines a series of animal experiments suggested by the simulations, and algorithms that bear consideration for application in robotic systems. We feel that these are the most important contributions of the work. The final section of this chapter outlines fruitful paths for future modeling efforts.

The Models and their Contributions

The most salient result of the modeling work is that we have demonstrated ways in which multiple depth cues can be used to enhance the speed and reliability of depth perception. Motivated by data from the literature on frogs and toads, we chose binocular disparity and lens accommodation as the two sources of depth cues for our study. To our knowledge, this is the first time that models of depth perception have been constructed using these two cues together.

The best way to use these two depth cues is to exploit the differences in their characteristics, rather than to simply take a weighted average of separate depth estimates based on each cue. Depth cues from binocular disparity are very precise, but suffer from inherent ambiguity; it is often difficult to determine which of several cues is the correct one. On the other hand, depth cues from accommodation are much less precise, and are subject to variation in accuracy with change in pupil diameter, but they are not ambiguous. Both of our models capitalize on the accuracy of binocular cues and the lack of ambiguity in the accommodation cues. Accommodation cues are used, in the models, to bias binocular cues so that their ambiguity is removed. They are not used (or are used with a low weight compared with binocular cues) to compute the final depth estimate. In this way the full binocular accuracy is maintained.

Our models demonstrate two different ways in which accommodation biasing can be done. In the "cue interaction model" of Chapter 3 a passive process is envisioned. A record of image intensity as a function of lens setting is maintained at each retinal position. This record provides one of the inputs to the model. The other input is a map of matches between the binocular images over a range of disparities. These inputs are used to drive two separate processes that share information so that they build nearly identical depth maps of the binocular visual field. We say that the model is passive because it takes no special actions based on the visual input that it receives. Its structure is such that its depth mapping activity is the same no matter what is going on in front of the eyes. By contrast, the "prey localization model" of Chapter 4 is based on an active process that adjusts the state of lens accommodation in direct response to the pattern of visual input. In this model, prey selectors on each side of the visual system supply the accommodation controller with the retinal coordinates of a selected object. The controller then adjusts the lens based on the disparity between the object's positions on the right and left retinas. This mechanism acts to reject binoculary mismatched selections. A mismatch will cause a severe defocusing of the lenses. This defocusing will alter the input to the prey selectors, eventually correcting the mismatch. Correct binocular matches will improve lens focus, thus enhancing the inputs to the prey selectors, and confirming the match.

In our modeling work we came to the conclusion that depth perception, at least in frogs and toads, is probably not a single process. Our current point of view is that since different visuo-motor tasks require different depth information, there is no reason to assume *a priori* that there is only one depth perception system.

The two different models are each specialized for providing the depth information necessary to support a particular visuo-motor activity. The "cue interaction model" is well suited to tasks involving navigation through the environment, and the "prey localization model" is especially appropriate for prey catching. The passive process of the "cue interaction model" is slower than the active process of the "prey localization model," but it produces a complete depth map. Navigation appears to require a complete depth map, but is generally concerned only with stationary objects, so that this map can be built using a relatively slow process. Prey catching, on the other hand, requires the location of only a single point, but the location of that point must be obtained rapidly and with high reliability. The active process of the "prey localization model" is well suited to this task.

If lens accommodation is used in conjunction with binocularity for depth perception, and if depth perception processes are specialized to support different kinds of activity then there must be a high degree of integration between the visual and motor systems. That vision should have a strong effect on motor activity comes as no surprise, but what has received very little attention is the effect that motor activity may have on vision. The extended model of accommodation control, described in Chapter 5, is an initial attempt at addressing some of the complexities of the interactions between the visual and motor systems.

Suggestions for Further Experiments with Frogs and Toads

The studies done with computer simulations of the models led to several interesting predictions, and also raised several important questions. The following discussion suggests experiments to both test the predictions and resolve the questions.

Our simulation of the "cue interaction model" indicates that accurate mapping of the depth of barrier-like objects will be dependent upon a high degree of correspondence between monocular and binocular depth cues. This interdependency is much less marked when very sparse input from prey-like objects is used. However, the only experimental data showing the interaction of binocularity and accommodation come from prey-catching studies. The sensitivity to mismatch between cues, in determining barrier depth, suggests that placing either lenses or prisms in front of the eyes of a frog or toad will greatly disturb its depth estimates for navigation. Thus, a useful set of behavioral experiments would be to observe toads, with lenses or prisms mounted in front of their eyes, negotiating a fence-like barrier. These observations could be analyzed to look for differences in depth-mediated behavior between control animals, animals with lenses, and animals with prisms. It would be interesting to see if the complex perturbations of the estimates of the depth of barriers noted in our simulation are exhibited by frogs or toads. The simulation predicts that concave lenses will have a consistent effect, with stronger lenses causing fences to appear closer to the animal than they do when weaker lenses are used. Prisms, on the other hand, should have a more variable effect, even causing the fence to appear fragmented, so that the animal might occassionally behave as if there were a gap in the fence. To date, experiments of this sort have not been attempted due to the difficulty of motivating toads to negotiate barriers under the necessary experimental conditions [Collett, personal communication].

The simulation of the "cue interaction model" takes considerably longer to converge when binocular cues are removed, leaving only accommodation cues. Although we have no simulation results to show this, it can be inferred that the extended accommodation control system of Chapter 5 would also be slower with binocular input removed, since this controller depends on binocular input to achieve a rapid approximate lens setting. Thus, we hypothesize that monocular blinding of frogs and toads will increase their snapping latency. Experiments to test this hypothesis should be easy to perform.

The simulation of the "prey localization model" showed a marked increase in convergence time when either simulated lenses or prisms were used. This is because the total gain of the system is reduced when binocular and monocular cues are out of correspondence. Thus, a good test of the way in which we envision that binocular and accommodation cues interact for prey catching would be to conduct timed prey-catching experiments with frogs or toads to see if they exhibit an increase in snapping latency when lenses or prisms are used.

The structures of both the "cue interaction" and the "prey localization" models are such that they would be much more responsive to lateral movements than to movements away from or towards the animal. The "cue interaction model" computes the spatial position of an object by using retinal position to index a calculated depth-segmentation of the visual field. Since small lateral movements would have only an angular effect, recomputation of spatial location would be merely a matter of changing the retinal position used to index the segmentation. A similar result obtains for the "prey localization model." This is because lateral movements would require only small adjustments in lens accommodative state, as opposed to the much larger adjustments that would be required to track an object moving towards or away from the eyes. Thus, we predict that frogs and toads should be much more responsive and accurate in compensating for lateral movement than for other movements. An interesting comparison may be made between this prediction and the results of Westheimer and McKee [1978] who found that stereoscopic depth perception in humans is considerably more accurate when an object is moved across the visual field than when its movement is in depth.

The "prey localization model" is based on the use of accommodation to disambiguate binocular depth cues, but ambiguity is only a problem when more than one prey object is present in the visual field. It is not possible to cause this model to make mismatch errors when only one prey object is presented. Therefore, a definitive test of the way in which the "prey localization model" utilizes monocular depth cues would be to observe prey-catching behavior in the presence of multiple stimuli in animals whose accommodative mechanism has been disabled. Jordan et al. [1980] have shown that binocular toads whose accommodation muscles have been treated with either a muscle relaxor or a muscle contractor have little trouble estimating the depth of single targets presented in the frontal binocular field. However, we predict that if similarly treated animals are presented with multiple targets they will exhibit a marked tendency toward making binocular correspondence errors, and will often snap at "ghost" rather than real targets.

To move beyond our preliminary models will require data from further animal experiments. The literature on depth perception in frogs and toads is quite rich when it comes to the perception of the depth of prey. However, although there is ample evidence that these animals are able to accurately judge the depth of stationary barrier-like objects [Collett, 1982], very little is known about the mechanisms underlying this form of depth resolution. It is quite certain that frogs and toads determine the depth of prey using both binocularity and lens accommodation, but the depth cues used in barrier depth estimation are as yet unknown. Another gap in the literature concerns the lens accommodation mechanism itself. There are, as yet, no data available showing the accommodation process in action. In Chapter 5, our model of accommodation control was built to address overt behavioral data, but a truly accurate description of this control will require data showing the accommodative response of the lenses when various visual configurations are presented.

Experiments with frogs or toads should be performed to test the hypothesis that both binocular and image focus cues are utilized to control lens accommodation. If a means were available to directly measure the accommodative state of the lenses then a test could be made to see if prisms have any effect upon this state. If there is an effect from prisms then it will be shown that binocular depth cues are being utilized. Also, if binocular cues are used to control accommodation then placing lenses in front of the eyes of binocular frogs or toads will have a much smaller effect on accommodation than a lens placed in front of the eye of a monocular animal.

To develop a more definitive model of lens accommodation than we have provided in Chapter 5 it will be necessary to have data to answer the following questions in both binocular and monocular animals:

1) When there is no stimulus present in the visual field does the lens scan in and out as if "searching" for a stimulus, or does it stay in a rest position?
2) When there are several barrier surfaces in the visual field does the lens intermittently focus on each of the surfaces in an exploratory way, or is the behavior similar to that when there is no stimulus?
3) When there is a prey and no other stimulus in the visual field, how does the accommodation process differ from that observed when there is no stimulus?
4) When there are barrier surfaces in the visual field and a prey is introduced, how does the accommodation process differ from that observed when there are only barriers?

To provide answers to these questions an experimental methodology must be developed that will allow the observation of the movement of the lens, or alternatively, the tension in the accommodation muscles, as the animal is confronted with different visual configurations.

The model of accommodation control presented in Chapter 5 is too tentative to allow us to make firm predictions about the probable outcome of the experiments suggested above, but we can list findings that would be confirmatory to its structure. If our model is correct then when confronted with a scene involving barriers but no prey stimuli, toads will scan their lenses in an oscillatory manner. Further, this oscillatory scanning will be momentarily interrupted as each barrier is brought into clear focus. If focus cues control the lenses separately for each eye then the accommodative state at which the two lenses stop will differ if the scene is bilaterally asymmetric. When a prey-like stimulus is introduced into the binocular visual field our model envisions that both lenses will rapidly accommodate to a state near to the depth of the prey, and then enter a slower fine-focusing phase.

Suggestions for Robotic Algorithms

Both depth perception models differ from previous theoretical models in that they involve adjusting the image sensors to obtain depth cues. For this reason, our models were not tested using static stereoscopic images, but instead were tested against an input scene that included the depth dimension. In this sense, both models are more suitable than models designed to process static images for representing the depth resolution process of animals. For this same reason, the models are also more appropriate for application in robotics. Robots interact not with static images but with the three-dimensional world, and thus could benefit from the greatly enhanced speed and reliability obtained by using multiple depth cues, some of which are obtained from sensor adjustment.

The method of using accommodation cues to disambiguate binocular cues that is outlined by the "cue interaction model" might be useful in robotic systems where the primary visual task is to maintain a map of the stationary objects in the visual environment. Such a scheme could employ a slow oscillatory scanning of the lenses of two imagers to provide image focus input, and disparity matching to provide binocular cues. Because of its organization, the algorithm underlying the "cue interaction model" would be very fast if implemented in an array processor.

The "prey localization model" has even more obvious applications in robotics. Given a limited class of objects to localize and a controlled visual environment, this model could be used as the basis for a simple but powerful object localization scheme. Its internal structure would allow easy implementation in parallel hardware for real-time performance. Additionally, its layers could be programmed as separate modules running independently in separate computing units. These modules could then be individually tuned to suit a wide variety of applications. For instance, the pattern recognizers could be tailored to recognize application-specific objects without modifying either the imagers, the selectors, or the accommodation controller.

A side benefit that would come from implementing the models in a robotic system would be to provide data that would help to further refine the models. The models have, as yet, been tested only on simple simulated two-dimensional scenes. Implementing them to interact with a three-dimensional environment and with direct video input is bound to uncover complications and possibilities that were not envisioned in our study.

Future Modeling Work

An obvious major modeling effort left undone by this research is to simulate the extended accommodation control system of Chapter 5, so that its ability to support our two depth models can be verified. However, the inherent complexity of this controller warrants more realistic testing than can be achieved with only simulated visual input. We plan to equip a laboratory to enable testing with real-time input from video cameras whose lenses are under computer control.

Another direction that we plan to take is to extend our efforts beyond the problem of depth perception to investigate the orientation and detour behavior of frogs and toads. We have already made some progress in this direction and have reported our initial results in Arbib and House [1982]. However, this early work was done before the development of the "prey localization model" and needs to be updated to address our current, more sophisticated, view of depth perception.

The work presented in this dissertation is only the beginning of an active collaboration between modelers and experimentalists in the investigation of an important aspect of the visuo-motor functioning of frogs and toads: depth perception. We strongly hope that our efforts will provide some guidance for fruitful experimental studies. We expect, in turn, to have to respond to new experimental findings by modifying and improving our models. With all good fortune this cooperative process will converge.

BIBLIOGRAPHY

Amari, S. and Arbib, M.A. (1977) Competition and cooperation in neural nets. In *Systems Neuroscience* (J. Metzler, Ed.), pp. 119-165, New York: Academic Press.

an der Heiden, U., Roth, G. (1983) A mathematical network model for retino-tectal prey recognition in amphibians. In *Proceedings of the Second Workshop on Visuomotor Coordination in Frog and Toad: Models and Experiments*, Technical Report 83-19, Computer and Information Science Dept., Univ. of Massachusetts, Amherst.

Arbib, M.A. (1981) Perceptual structures and distributed motor control. In *Handbook of Physiology -- The Nervous System II. Motor Control* (V.B. Brooks, Ed.), Bethesda: Amer. Physiological Society, pp. 1449-1480.

Arbib, M.A., House, D.H. (1983) Depth and detours: towards neural models. In *Proceedings of the Second Workshop on Visuomotor Coordination in Frog and Toad: Models and Experiments*, Technical Report 83-19, Computer and Information Science Dept., Univ. of Massachusetts, Amherst.

Autrum, H. (1959) Das fehlen unwillkürlicher augenbewegungen beim frosch. *Naturwissenschaften*, 46: 435.

Brown, W. T., Marks, W. B. (1977) Unit responses in the frog's caudal thalamus. *Brain, Behav. Evol.*, 14: 274-297.

Cervantes, F., Lara, R., Arbib, M. A. (1983) A neural model subserving prey-predator discrimination and size preference in anuran amphibia. In *Proceedings of the Second Workshop on Visuomotor Coordination in Frog and Toad: Models and Experiments*, Technical Report 83-19, Computer and Information Science Dept., Univ. of Massachusetts, Amherst.

Collett, T. (1977) Stereopsis in toads. *Nature*, 267: 349-351.

Collett, T. (1982) Do toads plan routes? A study of the detour behavior of *Bufo viridis*. *J. comp. Physiol.*, 146: 261-271

Collett, T., Harkness, L. I. K. (1982) Depth vision in animals. In *Analysis of Visual Behavior* (D.J. Ingle, M.A. Goodale and R.J.W. Mansfield, Eds.), Cambridge: MIT Press, pp. 111-176.

Collett, T., Udin, S. B. (1983) The role of the toad's nucleus isthmi in prey-catching behavior. In *Proceedings of the Second Workshop on Visuomotor Coordination in Frog and Toad: Models and Experiments*, Technical Report 83-19, Computer and Information Science Dept., Univ. of Massachusetts, Amherst.

Corvaja, N., d'Ascanio, P. (1981) Spinal projections from the mesencephalon in the toad. *Brain, Behav. Evol.*, 19: 205-213.

Cromarty, A., Sutton, R. (1983) *The GUS Device-Independent Graphics System Reference Manual.* Online documentation, Department of Computer & Information Science, University of Massachusetts, Amherst. Unpublished.

Dev, P. (1975) Perception of depth surfaces in random-dot stereograms: a neural model. *Int. J. Man-Machine Studies*, 7: 511-528.

Didday, R. L. (1970) *The Simulation and Modelling of Distributed Information Processing in the Frog Visual System.* Doctoral dissertation, Stanford University.

Didday, R.L. (1976) A model of visuomotor mechanisms in the frog optic tectum. *Math. Biosci.*, 30: 169-180.

Enroth-Cugell, C., Robson, J. G. (1966) The contrast sensitivity of retinal ganglion cells of the cat. *J. Physiol.*, 187: 517-552.

Epstein, S. (1979) *Vermin Users Manual.* Department of Computer & Information Science, University of Massachusetts, Amherst. Unpublished project report.

Ewert, J.-P. (1971) Single unit response of the toad (*Bufo americanus*) caudal thalamus to visual objects. *Z Vergl. Physiol*, 74: 81-102.

Ewert, J.-P. (1976) The visual system of the toad: Behavioral and physiological studies on a pattern recognition system. In *The Amphibian Visual System A Multidisciplinary Approach* (K. Fite, Ed.), New York: Academic Press, pp. 141-202.

Ewert, J.-P. (1980) *Neuroethology*, Berlin: Springer-Verlag, pp. 69-76.

Ewert, J.-P. (1982) Neuronal basis of configurational prey selection in the common toad. In *The Analysis of Visual Behavior* (D. Ingle, M. Goodale, and R. Mansfield, Eds.), Cambridge: MIT Press, pp. 7-45.

Ewert, J.-P., von Seelen (1974) Neurobiologie und system-theorie eines visuellen muster-erkennungsmechanismus bei krote. *Kybernetik*, 14: 167-183.

Finch, D. J., Collett, T. S. (1983) Small-field, binocular neurons in the superficial layers of the frog optica tectum. *Proc. R. Soc. Lo.* B, 217: 491-497.

Fischer, B. (1973) Overlap of receptive field centers and representation of the visual field in cat's optic tract. *Vis. Res.*, 13: 2113-2120.

Fite, K. V. (1969) Single-unit analysis of binocular neurons in the frog optic tectum. *Exp. Neurol.*, 24: 475-486.

Fite, K. V., Scalia, F. (1976) Central visual pathways in the frog. In *The Amphibian Visual System A Multidisciplinary Approach* (K. Fite, Ed.), Academic Press, pp. 87-118.

Frisby, J. P., Mayhew, J. E. W. (1980) Spatial frequency tuned channels: Implications for structure and function from psychophysical and computational studies of stereopsis. *Phil. Trans. R. Soc. Lo. B*, 290: 95-116.

Gaillard, F., Galand, G., (1980) A possible neurophysiological basis for depth perception in frogs: existence of a horopter surface. *J. Physiol., Paris*, 76: 123-127.

Georgopoulis, A. P., Caminiti, R., Kalaska, J. F., Massey, J. T. (1983) Spatial coding of movement: a hypothesis concerning the coding of movement direction by motor cortical populations. In *Neural Coding of Motor Performance* (W. Masson, J. Paillard, W. Schultz, M. Wiesendanger, Eds.), Berlin: Springer Verlag, pp. 327-336.

Glasser, S., Ingle, D. J. (1978) The nucleus isthmus as a relay station in the ipsilateral visual projection to the frog's optic tectum. *Brain Res.*, 159: 214-218.

Grobstein, P., Comer, C., Hollyday, M., Archer, S. M. (1978) A crossed isthmo-tectal projection in *Rana pipiens* and its involvement in the ipsilateral visuotectal projection. *Brain Res.*, 156: 117-123.

Grobstein, P., Comer, C., Kostyk, S. (1980) The potential binocular field and its tectal representation in *Rana pipiens*. *J. Comp. Neurol.*, 190:175-185.

Grobstein, P., Comer, C. (1983) The nucleus isthmi as an intertectal relay for the ipsilateral oculo-tectal projection in the frog, *Rana pipiens*. *J. Comp. Neurol.*, 217: 54-74.

Grobstein, P., Comer, C., and Kostyk, S.K. (1983) Frog prey capture behavior: between sensory maps and directed motor output. In *Advances in Vertebrate Neuroethology* (J.-P. Ewert, R.R. Capranica and D.J. Ingle, eds.), London: Plenum Press, pp. 331-347.

Gruberg, E. R., Udin, S. B. (1978) Topographic projections between the nucleus isthmi and the tectum of the frog *Rana pipiens*. *J. Comp. Neurol.*, 179: 487-500.

Gruberg, E. R., Lettvin, J. Y. (1980) Anatomy and physiology of a binocular system in the frog *Rana pipiens*. *Brain Res.*, 192: 313-325.

Grüsser, O.-J., Grüsser-Cornehls, U. (1976) Neurophysiology of the anuran visual system. In *Frog Neurobiology A Handbook* (R. Linás, W. Precht, Eds.), Berlin: Springer-Verlag, pp. 297-385.

Hirai, Y., Fukushima, K. (1978) An inference upon the neural network finding binocular correspondence. *Biol. Cybernetics*, 31: 209-217.

Ingle, D. (1968) Visual releasers of prey catching behavior in frogs and toads. *Brain, Behav. Evol.*, 1: 500-518.

Ingle, D. (1970) Visuomotor functions of the frog optic tectum. *Brain, Behav. Evol.*, 3: 57-71.

Ingle, D. (1973) Selective choice between double prey objects by frogs. *Brain, Behav. Evol.*, 7: 127-144.

Ingle, D. (1976) Spatial visions in anurans. In *The Amphibian Visual System a Multidisciplinary Approach* (K. Fite, Ed.), New York: Academic Press, pp. 119-140.

Ingle, D. (1977) Detection of stationary objects by frogs (Rana pipiens) after ablation of the optic tectum. *J. Comp. Physiol. Psychol.*, 391: 1359-1364.

Ingle, D. (1980) Some effects of pretectum lesions on the frog's detection of stationary objects, *Behav. Brain Res.*, 1: 139-163.

Ingle, D. (1982) The organization of visuomotor behaviors in vertebrates. In *The Analysis of Visual Behavior* (D. J. Ingle, M. Goodale, and R. Mansfield, Eds.), Cambridge: MIT Press, pp. 67-109.

Ingle, D. (1983) Visual mechanisms of optic tectum and pretectum related to stimulus localization in frogs and toads. In *Advances in Vertebrate Neuroethology* (J.-P. Ewert, R.R. Capranica and D.J. Ingle, eds.), London: Plenum Press, pp. 177-226.

Jordan, M., Luthardt, G., Meyer-Naujoks, Chr., Roth, G. (1980) The role of eye accommodation in the depth perception of common toads. *Z. Naturforsch.*, 35c: 851-852.

Julesz, B. (1971) *Foundations of Cyclopean Perception*, University of Chicago Press.

Kicliter, E. (1973) Flux, wavelength and movement discrimination in frogs: forebrain and midbrain contributions. *Brain, Behav. Evol.*, 8: 340-365.

Kostyk, S., Grobstein, P. (1982) Visual orienting deficits in frogs with various unilateral lesions. *Behav. Brain Res.*, 6: 379-388.

Lara, R., Arbib, M.A. (1982) A neural model of interaction between tectum and pretectum in prey selection. *Cognition and Brain Theory*, 5: 149-171.

Lara, R., Arbib, M. A., Cromarty, A. S. (1982) The role of the tectal column in facilitation of amphibian prey-catching behavior. *Biological Cybernetics*, 44: 185-196.

Lara, R., Carmona, M., Daza, F., Cruz, A. (1983) A global model of the neural mechanisms responsible for visuomotor coordination in toads. In *Proceedings of the Second Workshop on Visuomotor Coordination in Frog and Toad: Models and Experiments*, Technical Report 83-19, Computer and Information Science Dept., Univ. of Massachusetts, Amherst.

Lawton, D. T. (1984) *Processing Dynamic Image Sequences from a Moving sensor.* Doctoral dissertation, University of Massachusetts, Amherst.

Lettvin, J. Y., Maturana, H. R., McCulloch, W. S., Pitts, W. H. (1959) What the frog's eye tells the frog's brain. *Proc. Instn elect. Engrs*, 47: 1940-1951.

Ligthart, G., Groen, C. A. (1982) A comparison of different autofocus algorithms. *Proc. 6th Int. Conf. on Pattern Recognition*, Oct. 19-22: 597-600.

Lock, A., Collett, T. (1979) A toad's devious approach to its prey - a study of some complex uses of depth vision. *J. Comp. Physiol.*, 131: 179-189.

Lock, A., Collett, T. (1980) The three - dimensional world of a toad. *Proc. R. Soc. Lo. B*, 206: 481-487.

Marr, D., Poggio, T. (1976) Cooperative computation of stereo disparity. *Science*, 194: 283-287.

Marr, D., Poggio, T. (1979) A computational theory of human stereo vision. *Proc. R. Soc. Lo. B*, 204: 301-328.

McIlwain, J. T. (1982) Lateral spread of neural excitation during microstimulation in intermediate gray layer of cat's superior colliculus. *J. Neurophysiol.*, 47: 167-178.

Montgomery, N., Fite, K., Taylor, M., Bengston, L. (1982) Neural correlates of optokinetic nystagmus in the mesencephalon of *Rana pipiens*: a functional analysis. *Brain Behav. Evol.*, 21: 137-150.

Morse, A. C. (1979) *What the Grinnell Graphics Routines are and How to Use Them.*

Online documentation, Department of Computer & Information Science, University of Massachusetts, Amherst. Unpublished.

Morse, A. C. (1981) *Using Computer Graphics as an Aid to Understanding Simulation Data*. Doctoral dissertation, University of Massachusetts, Amherst.

Muntz, W. R. A. (1962) Microelectrode recordings from the diencephalon of the frog (*Rana pipiens*) and a blue-sensitive system. *J. Neurophysiol.*, 25: 699-711.

Neisser, U. (1976) *Cognition and Reality*, San Francisco: W.H. Freeman and Co., pp. 20-24.

Overton, K. J. (1980) Design and use of a computer controlled movie camera facility. Technical Report 80-08, Computer and Information Science Dept., Univ. of Massachusetts, Amherst.

Panum, P. L. (1858) *Physiologische Untersuchungen über das Sehen mit Zwei Augen*, Kiel: Homann.

Pitts, W. H., McCulloch, W. S. (1947) How we know universals: the perception of auditory and visual forms. *Bull. Math. Biophys.*, 9: 127-147.

Podufal, G. (1971) *Zur Entfernungsmessung und Grossenbeurteilung Durch die Erdkröte (B. bufo, L.)*. Doctoral dissertation, University of Göttingen.

Poggio, G. F. (1979) Mechanisms of stereopsis in monkey visual cortex. *Trends in Neuroscience*, 2: 199-201.

Prager, J. M., Arbib, M. A. (1983) Computing the optic flow: the MATCH algorithm and prediction. *Computer Vision, Graphics, and Image Processing*, 24: 271-304.

Raybourn, M. S. (1975) Spatial and temporal organization of the binocular input to frog optic tectum. *Brain, Behav. Evol.*, 11: 161-178.

Riss, W. (1968) Comment on the article of D. J. Ingle titled 'Interocular integration of visual learning by goldfish'. *Brain, Behav. Evol.*, 1: 86-87.

Robinson, D. A. (1981) The use of control systems analysis in the neurophysiology of eye movements. *Ann. Rev. Neurosci.*, 4: 463-503

Rossel, S. (1980) Foveal fixation and tracking in the praying mantis. *J. Comp. Physiol.*, 139: 307-331.

Rubinson, K. (1968) Projections of the tectum opticum of the frog. *Brain, Behav. Evol.*, 1: 529-561.

Rubinson, K., Colman, D. R. (1972) Designated discussion: a preliminary report on ascending thalamic afferents in *Rana pipiens*. *Brain, Behav. Evol.*, 6: 69-74.

Scalia, F., Fite, K. (1974) A retinotopic analysis of the central connections of the optic nerve in the frog. *J. Comp. Neur.*, 158: 455-478.

Schipperheyn, J. J. (1963) Respiratory eye movements and perception of stationary objects in the frog. *Acta physiol. pharmacol. neerl.*, 12: 157-159.

Skarf, B., Jacobson, M. (1974) Development of binocularly driven single units in frogs raised with asymmetrical visual stimulation. *Exp. Neurol.*, 42: 669-686.

Selker, T. (1982) Image-based focusing. *Robotics and Industrial Inspection*, (D.P. Casasent, Ed.) Proc. SPIE, 360: 96-99.

Stevens, R. J. (1973) A cholinergic inhibitory system in the frog optic tectum: its role in visual electrical responses and feeding behavior. *Brain Res.*, 49: 309-321.

Trehub, A. (1978) Neuronal model for stereoscopic vision. *J. theor. Biol.*, 71: 479-486.

Wang, S.-J., Yan, K., Wang, Y.-T. (1981) Visual field topography and binocular responses in frog's nucleus isthmi. *Scientia Sinica*, 24: 1292-1301.

Westheimer, G., McKee, S. P. (1978) Stereoscopic acuity for moving retinal images. *J. Opt. Soc. Am.*, 68: 450-455.

APPENDIX A

MATHEMATICAL AND COMPUTER REPRESENTATION OF THE MODELS

The cooperative-competitive computational mechanism used in both the "cue interaction" and the "prey localization" models had its origin in an analytical paper by Amari and Arbib [1977]. This format was continued throughout this dissertation by choosing to represent both models by continuous equations that were then solved numerically, rather than starting with difference equations that could be solved directly. The continuous representation is based upon a definition of a single idealized neural unit. This definition is then extrapolated over a continuous homogeneous layer. Integro-differential equations are used to describe a layer's internal potential. In this appendix we explain the development of this general representation, and provide details of the computer simulations of both depth models.

Mathematical Representation of a Single Neural Unit

Our representation of a single neural unit consists of

1) a set of weighted inputs,
2) an internal potential,
3) a differential equation that relates change in internal potential to both the current potential and the strength of the inputs,
4) a function that converts internal potential to an external potential or firing rate.

We assume that the differential equation relating the inputs to the internal potential is both first order and linear, with time-constant T. Thus, the equation describing internal potential is

$$T\dot{M} = -M + \sum_{i=1}^{n} K_i I_i, \tag{A1}$$

where: M represents internal potential,

n is the number of inputs,

I_i, $1 \leq i \leq n$, is the ith input,

and K_i is the ith input weight.

Two different functions may be used to convert internal potential M to external firing rate. The first of these is a sigmoidal saturation-threshold function (Fig. 47a) and is used in the representation of excitatory units. Inhibitory units employ a simple thresholded linear function (Fig. 47b). The symbol f, with appropriate subscripts, is used throughout the dissertation to represent a saturation-threshold function and the symbol g to represent the thresholded linear function.

The general form of the saturation-threshold function used in all of the computer simulations is

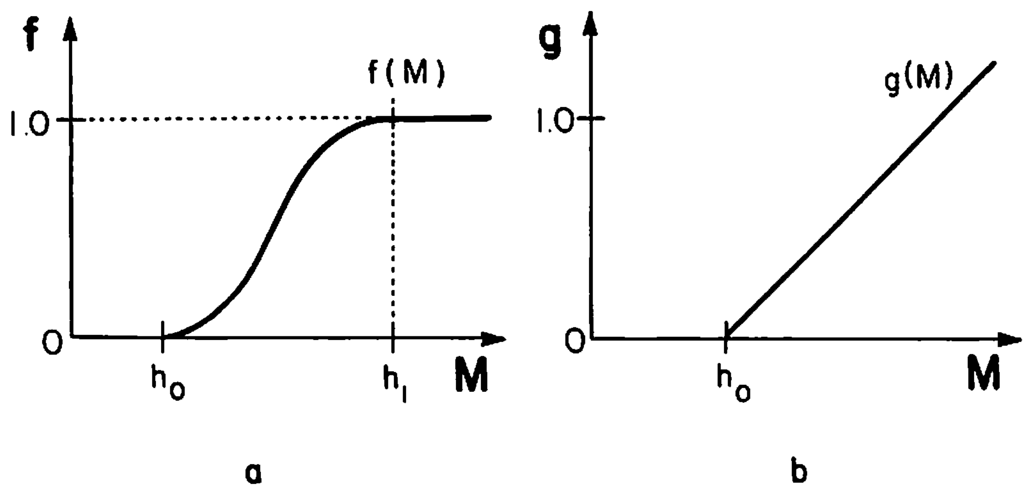


Fig. 47 - Threshold Functions.

$$f(M) = \begin{cases} 0, & M < h_0 \\ \sigma^2(3 - 2\sigma), & h_0 \le M \le h_1 \\ 1, & M > h_1, \end{cases} \qquad \text{(A2)}$$

where: h_0 is the threshold,

h_1 is the saturation level,

and $\sigma = (M - h_0) / (h_1 - h_0)$.

A particular instance of the saturation-threshold function is completely defined by specifying h_0 and h_1. The coefficients of the cubic polynomial that represents the function in the region between h_0 and h_1 were derived by constraining f and its first derivative to be continuous.

The thresholded linear function is completely defined by specifying its single threshold, h_0. Its formulation is given by

$$g(M) = \begin{cases} 0, & M < h_0 \\ M - h_0, & M \geq h_0. \end{cases} \tag{A3}$$

Mathematical Representation of a Neural Layer

A neural layer is a continuous approximation to an array of individual neurons. We construct such a layer by first replicating the definition of a neural unit over a continuous one or two-dimensional surface, and then defining the connectivity between loci on the layer. A layer is homogeneous in all of its properties. For a one-dimensional layer we let $w'(x,y)$ represent the weight of the influence of a cell at position y on a cell at position x. We constrain this weighting function by requiring that the strength of the connections between two points be a function only of the Euclidian distance between points x and y, i.e. $|x - y|$. Therefore, this connectivity may represented by an even function w, defined by

$$w(x-y) = w'(x,y),$$

with

$$w(s) = w(-s).$$

For a two-dimensional layer, connectivity between two points is constrained to be a function of the magnitudes of the two components r and s of the vector difference between the two points. Thus, connectivity is not radially symmetric but does have the property

$$w(r,s) = w(\pm r, \pm s).$$

If we let M(x,y,t) represent the internal point potential at position (x,y) and time t in two-dimensional layer M, and let f[M(x,y,t)] be the firing rate at that point, then the strength of the stimulation received by a point from its neighboring points is given by the convolution

$$\iint w(x - \zeta, y - \eta)\, f[M(\zeta,\eta,t)]\, d\zeta\, d\eta.$$

The complete description of the point potential in a two-dimensional layer is given by

$$\tau\, \dot{M}(x,y,t) = -M(x,y,t) + \iint w(x - \zeta, y - \eta)\, f[M(\zeta,\eta,t)]\, d\zeta\, d\eta + \sum_{i=1}^{n} K_i\, I_i(x,y,t), \tag{A4}$$

where: the variables $I_i(x,y,t)$ represent external input.

Similarly, a one-dimensional layer U is described by

$$\tau\, \dot{U}(x,t) = -U(x,t) + \int w(x - \zeta)\, f[U(\zeta,t)]\, d\zeta + \sum_{i=1}^{n} K_i\, I_i(x,t). \tag{A5}$$

The external firing rate at a point on a layer is given by either

$$f[M(x,y,t)] \quad \text{or} \quad g[M(x,y,t)].$$

The functions w that govern the lateral spread of excitation within a layer are represented in the simulations by piecewise polynomial functions. All of the spread functions used were of the form shown in Fig. 48. Neither of the models required the simulation of a two dimensional spread function.

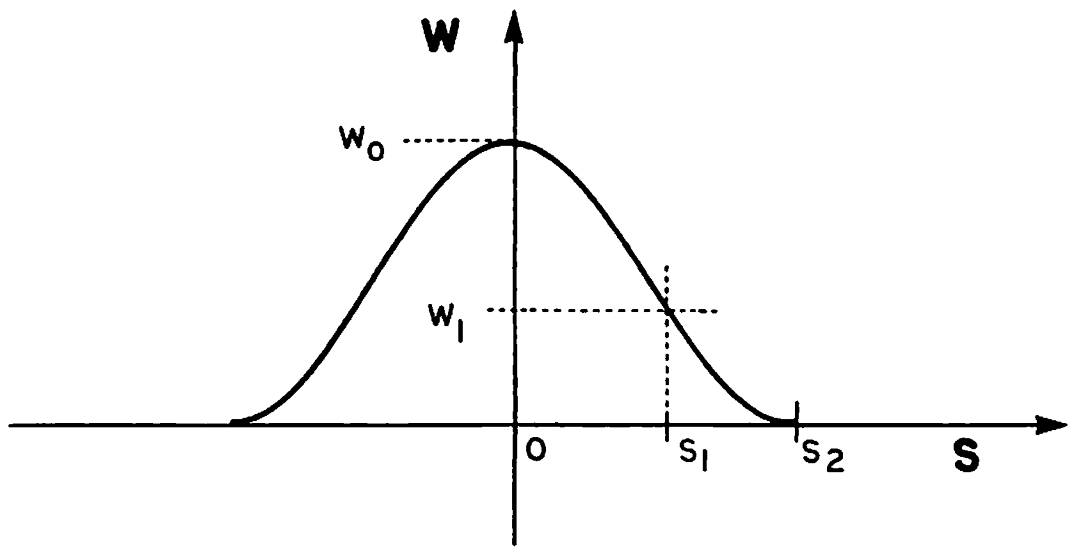


Fig. 48 - Typical Spread Curve.

In Fig. 48 the s axis represents the difference between the positions of two points. The points $(0,w_0)$, (s_1,w_1), and $(s_2,0)$ represent three knot points through which the piecewise polynomial is drawn. We represent the portion of the curve between 0 and s_1 by a quadratic and the portion between s_1 and s_2 by a cubic. We constrain the curve by requiring it and its first derivative to be everywhere continuous. The description of w is then

$$w(s) = \begin{cases} (w_1 - w_0)\,\sigma_1^2 + w_0, & 0 \le |s| \le s_1 \\ C_1\,\sigma_2^3 + C_2\,\sigma_2^2, & s_1 < |s| < s_2 \\ 0, & |s| \ge s_2, \end{cases} \tag{A6}$$

where: $\sigma_1 = |s| \,/\, s_1,$

$\sigma_2 = (s_2 - |s|) \,/\, (s_2 - s_1),$

$C_1 = w_1 - C_2,$

$C_2 = 3\,w_1 + 2\,(w_1 - w_0)\,(s_2 - s_1) \,/\, s_1.$

Thus, a particular instance of the spread function w is completely specified by the four parameters w_0, w_1, s_1, and s_2.

Numerical Methods

Both models are qualitative representations of computational processes and do not attempt to replicate the detailed functioning of known physical systems. Further, for the most part we were not interested in the transient performance of the simulations, but only in their equilibria. Therefore, we did not feel that the use of sophisticated numerical techniques was required. Instead, in the interests of simplicity and computational speed, we chose to use extremely simple approximation methods. The spatial integrals in the equations describing the models were computed using the trapezoidal rule, and the differential equations were solved using Euler's method. To insure that the use of Euler's method did not seriously degrade accuracy we chose the time increments for the simulations carefully. This was done by running the simulations against standard test scenes using various time increments, and noting the point at which reduction of the time increment ceased to have a noticable effect on the resulting equilibrium state. All simulations were begun from an initially inert state (zero potential in all layers), and excitation was taken to be zero beyond the boundaries of a layer when computing the spatial convolution integrals.

The Cue Interaction Model

In the computer simulation of the "cue interaction model" of Chapter 3 the monocular layer M and the stereoptic layer S were modeled as 41 x 41 arrays. The corresponding inhibitory layers U and V were modeled as vectors of length 41. The binocular disparity input plane D and the accommodation input plane A were also modeled as 41 × 41 arrays. These inputs were calculated as described in Appendix B and, when required, included the effect of simulated interposed lenses and prisms.

In order to establish a reasonable starting set of parameters for the simulation we conducted an equilibrium analysis. To begin this analysis, the time derivatives in eqs. (3) of Chapter 3 were set to zero to obtain the equilibrium equations

$$M(q,d) = \int w_m(q - \zeta)\, f[M(\zeta,d)]\, d\zeta + K_{sm}\, f[S(q,d)] - K_m\, g[U(q)] + K_a\, A(q,d),$$
$$U(q) = K_u \int f[M(q,\eta)]\, d\eta,$$

$$S(q,d) = \int w_s(q - \zeta)\, f[S(\zeta,d)]\, d\zeta + K_{ms}\, f[M(q,d)] - K_s\, g[V(q)] + K_d\, D(q,d),$$
$$V(q) = K_v \int f[S(q,\eta)]\, d\eta. \tag{A7}$$

These equilibrium equations were simplified by applying several *ad hoc* approximations and constraints. First, the equations were approximated by discrete equations, with indices i and j replacing spatial coordinates q and d, summations replacing integrals, and discrete steps Δq and Δd replacing the differentials. With the exception of the input gains K_a and K_d, corresponding gains and spread functions were constrained to be identical across the two pairs of equations. Cross-coupling gains K_{sm} and K_{ms} were replaced by K_c, inhibitory feedback gains K_m and K_s were replaced by K_i, and inhibitory layer input gains K_u and K_v were replaced by K_o. Similarly, spread functions w_m and w_s were replaced by the single function w. The extent of the summations used to approximate the spatial convolutions were constrained to three discrete spatial steps. This allowed the spread function w to be reduced to two constants W_0 and W_1, with W_0, the integral of w from $-\Delta d/2$ to $\Delta d/2$, representing the central portion of w; and W_1, the integral of w from $\Delta d/2$ to ∞, representing the two portions of w to either side of the central portion. The application of all of these simplifications to eqs. (A7) gives

$$
\begin{aligned}
M(i,j) &= W_0 f[M(i,j)] + W_1\{f[M(i-1,j)] + f[M(i+1,j)]\} + K_c f[S(i,j)] - \\
&\qquad K_i g[U(i)] + K_a A(i,j), \\
U(i) &= K_0 \Delta d \sum_k f[M(i,k)], \\
S(i,j) &= W_0 f[S(i,j)] + W_1\{f[S(i-1,j)] + f[S(i+1,j)]\} + K_c f[M(i,j)] - \\
&\qquad K_i g[V(i)] + K_d D(i,j), \\
V(i) &= K_0 \Delta d \sum_k f[S(i,k)].
\end{aligned}
\tag{A8}
$$

The analysis was further simplified by *a priori* selection of the saturation level of function f, and the threshold of function g. We let h represent the threshold of f and defined its saturation level to be h + 1. The threshold of function g was defined to be zero.

Assumptions were also made to constrain the form of the equilibrium solutions. First, we invoked the uniqueness constraint that, for each discrete retinal position i, at most one element of array M and one element of array S will have a level of excitation above threshold h. Further, we sought to enforce the constraint that if an element M(i,j) or S(i,j) were above threshold, that it would be in saturation; and also sought to require that if a point in one excitatory layer, for example M(i,j), were above threshold, then the corresponding point in the other layer, S(i,j), would also be above threshold. These constraints are summarized in Table 2. It follows from these constraints that for a given retinal position index i, either no points of arrays M and S are above threshold h, or exactly one point is in saturation in each array and that these points have the same disparity index j.

Table 2 - Equilibrium Solution Constraints.

1) $M(i,j) > h \rightarrow M(i,k) < h, \; k \neq j$

   $S(i,j) > h \rightarrow S(i,k) < h, \; k \neq j$

2) $M(i,j) > h \rightarrow M(i,j) \geq h + 1$

   $S(i,j) > h \rightarrow S(i,j) \geq h + 1$

3) $M(i,j) > h \rightarrow S(i,j) > h$

   $S(i,j) > h \rightarrow M(i,j) > h$

It also follows from the constraints of Table 2 and the definition of saturation function f that we may define a parameter

$$\Omega_{i,j} = f[M(i-1,j)] + f[M(i+1,j)] = f[S(i-1,j)] + f[S(i+1,j)],$$

that will take on values 0, 1, or 2 depending upon how many left and right neighbors of point (i,j) are above threshold. To take advantage of this, the continuity constraint of Chapter 3 was invoked in a very strict sense: if a point is in saturation then its neighbors are in saturation, and if it is not above threshold then its neighbors will not be above threshold. Thus we have

$$\Omega_{i,j} = \begin{cases} 2, & M(i,j) > h + 1 \text{ and } S(i,j) > h + 1 \\ 0, & M(i,j) < h \text{ and } S(i,j) < h. \end{cases} \quad \text{(A9)}$$

Because of the three constraints listed in Table 2 and the choices made for parameters of threshold functions f and g, the equilibrium equations may take two possible forms. If there is a disparity index j at retinal-position i for which M and S are above threshold then

$$
\begin{aligned}
M(i,k) &= \begin{cases} W_0 + 2W_1 + K_c - K_iK_o\Delta d + K_aA(i,k), & k = j \\ - K_iK_o\Delta d + K_aA(i,k), & k \neq j \end{cases} \\
U(i) &= K_o\Delta d, \\
S(i,k) &= \begin{cases} W_0 + 2W_1 + K_c - K_iK_o\Delta d + K_dD(i,k), & k = j \\ - K_iK_o\Delta d + K_dD(i,k), & k \neq j \end{cases} \\
V(i) &= K_o\Delta d.
\end{aligned}
\tag{A10}
$$

If, however, no point at retinal-position index i is above threshold then

$$
\begin{aligned}
M(i,j) &= K_aA(i,j), \\
U(i) &= 0, \\
S(i,j) &= K_dD(i,j), \\
V(i) &= 0.
\end{aligned}
\tag{A11}
$$

Eqs. (A10) together with the various constraints serve to place bounds on the allowable parameter settings. In order for both the uniqueness constraint 1) and the saturation constraint 2) to hold we must have

$$W_0 + 2W_1 + K_c - K_iK_o\Delta d + K_aA(i,j) > h + 1,$$

$$W_0 + 2W_1 + K_c - 2K_iK_o\Delta d + K_aA(i,j) < h.$$

These inequalities may be subtracted to yield the restriction on net inhibitory gain

$$K_i K_o \Delta d > 1. \qquad \text{(A12)}$$

In order for the similarity constraint 3) and the saturation constraint 2) to hold we must have

$$W_0 + 2W_1 + K_c - K_i K_o \Delta d + K_a A(i,j) > h + 1,$$
$$W_0 + 2W_1 - K_i K_o \Delta d + K_a A(i,j) < h.$$

These inequalities may be subtracted to yield the restriction on the cross-coupling gain

$$K_c > 1. \qquad \text{(A13)}$$

In order for both the continuity constraint and the saturation constraint 2) to hold we must have

$$W_0 + 2W_1 + K_c - K_i K_o \Delta d + K_a A(i,j) > h + 1,$$
$$W_0 + K_c - 2K_i K_o \Delta d + K_a A(i,j) < h.$$

These inequalities may be subtracted to yield the restriction on the non-central portion of the excitatory spread function

$$W_1 > 0.5. \qquad \text{(A14)}$$

By adding an additional constraint that the model be able to work in a solely monocular mode, we obtain the other needed restriction on the excitatory spread function. Setting K_d to zero in eq. (A10) and invoking the similarity constraint 3) and the saturation constraint 2) gives

$$W_0 > h + 1 + K_i K_o \Delta d - K_c - 2W_1. \tag{A15}$$

The choice of input gains K_a and K_d were restricted by applying a further constraint that there be threshold input values A_0 and D_0 such that

$$M(i,j) < h, \text{ for all } j \rightarrow A(i,j) < A_0, \text{ for all } j,$$

and

$$S(i,j) < h, \text{ for all } j \rightarrow D(i,j) < D_0, \text{ for all } j.$$

Eqs. (A11) hold only when all elements of M and S at retinal position i are below threshold h, so together with the input threshold constraint they require

$$\begin{aligned} K_a &< h \,/\, A_0, \\ K_d &< h \,/\, D_0. \end{aligned} \tag{A16}$$

It is clear that not all of the constraints applied in the derivation of inequalities (A12)–(A16) can hold everywhere. However, they served to simplify the analysis enough to allow us to make an educated guess at a reasonable set of initial parameter settings. The most important constraint is uniqueness 1), so we chose the inhibitory gain to be 80% greater than the lower limit given by (A12). The maximum allowable disparity represented in the simulation was ±0.75, and the disparity coordinate was represented by 41 discrete steps (indices −20 to 20) so that the disparity step size Δd was 0.0375. The inhibitory layer input gain K_o was set to 80.0, and the inhibitory feedback gain K_i was set to 0.6. This gave a net inhibitory gain $K_iK_0\Delta d$ of 1.8. The similarity and continuity constraints were less important. They were weakly enforced by setting the cross-coupling gain and the non-central portion of the excitatory spread function to the borderline values of inequalities (A13) and (A14): K_c was set to 1.0, and W_1 was set to 0.5. The threshold h was arbitrarily chosen to be 0.1. After making these choices, the resulting borderline value of inequality (A15) was used to set the central portion W_0 of the excitatory spread function to 0.9. Finally, the simulation was made more sensitive to binocular input than to monocular input by setting D_0, the threshold on disparity input, to 0.2 and A_0, the threshold on accommodation input, to 0.5. The borderline values of inequalities (A16), were used to set the disparity input gain K_d to 0.5 and and the accommodation input gain K_a to 0.2.

The equilibrium analysis of the "cue interaction model" gave us a starting set of simulation parameters. The nominal parameter settings used in the experiments were determined by making adjustments to this starting set both to improve transient performance and to better approximate behavioral data. These nominal parameter settings are shown in Table 3. The only parameters requiring adjustment were the cross-coupling gain K_c, which was reduced 20% to 0.8, and the excitatory spread function, whose central portion W_0 was reduced 25% to 0.68 and whose non-central portion was reduced 50% to 0.25.

Table 3 - Cue Interaction Model Simulation Parameter Settings.

Simulation Time Increment, $\Delta t = 0.05$

Time Constants

$T_m = 0.30$ $T_u = 0.10$ $T_s = 0.30$ $T_v = 0.10$

Saturation Functions

	h_0	h_1
f	0.10	1.10
g	0.00	—

Spread Functions

	w_0	w_1	s_1	s_2	W_0	W_1
w_m	17.78	7.11	0.0375	0.0750	0.68	0.25
			(3.375°)	(6.750°)		
w_s	17.78	7.11	0.0375	0.0750	0.68	0.25
			(3.375°)	(6.750°)		

Gains

$K_{sm} = 0.80$ $K_m = 0.60$ $K_u = 80.0$ $K_a = 0.20$
$K_{ms} = 0.80$ $K_s = 0.60$ $K_v = 80.0$ $K_d = 0.50$

The Prey Localization Model

Most of the layers of the "prey localization model" of Chapter 4 are represented as one-dimensional homogeneous layers of the type described by eq. (A5). Exceptions to this are the inhibitory cell layers, that are modeled as single neural units as described in eq. (A1); and the imager layers that are simply the retinal inputs R_L and R_R described by eq. (B3) in Appendix B. This model also contains an accommodation controller that is modeled as a lumped parameter system whose input is two sets of vector quantities.

All of the one-dimensional layers of this model were represented in the computer simulation as vectors of length 41. The accommodative input planes A_L and A_R were represented by 41×41 arrays. These inputs were calculated from the retinal inputs R_L and R_R as described in Appendix B and, when required, included the effect of simulated inperposed lenses and prisms.

As with the "cue interaction model," an equilibrium analysis was conducted in order to establish a reasonable set of parameters for the simulation. This analysis was begun by setting the derivatives in eqs. (4)–(10) to zero to obtain equilibrium equations, and then making a discrete approximation to the continuous equations. In the resulting discrete equations, all layers are represented by arrays with index i replacing the retinal-position coordinate q, and index j replacing the disparity coordinate d. The equations were further simplified by making several approximations. Each spread function w was replaced by a constant W that represents the integral under the function w for one retinal-position increment Δq about its center. To justify this step it was assumed that both w_t and w_b are narrow with respect Δq, and that w_i, although it is broad with respect to Δq, is nearly constant over its width. The thresholds h_0 for the saturation-threshold functions f_t and f_b were arbitrarily set to 0, and the saturation levels h_1 were set to 1. The threshold for the threshold function g was set to 0. Also, the subscripts L and R, indicating the right and left sides of the visual system, were dropped since the analysis required only one set of equations.

Given these simplifications, the equilibrium solution to eq. (4), governing the pattern recognizers, is

$$T(i) = W_t f_t[T(i)] + K_{at} A(i,j).$$

We wanted the level of excitation T in the pattern-recognizers to exactly reach saturation with the accommodation input at its maximum value of 1.0. Thus, K_a and W_t are related by

$$K_{at} = 1 - W_t \qquad \text{(A17)}.$$

We chose a value of 0.75 for W_t and 0.25 for K_{at}.

The equilibrium solution to eq. (5), governing the binocular layer, is

$$B(i) = W_b f_b[B(i)] + 2I(i) + K_{tb} f_t[T(i)] - K_{ub} U.$$

To assure robust performance, we wanted the binocular layer to reach a value 80% above saturation level when receiving full input from both the pattern-recognizer T and the two relay layers I. We also wanted to assure that in equilibrium the binocular cross-coupling through the relay layers would dominate the direct input from the pattern-recognizer. Thus, we set K_{tb} to 1/3 of the total relay input giving

$$K_{tb} = 0.67\ I_{max}, \tag{A18}$$

where I_{max} is the maximum expected equilibrium level of the relay input. Applying these constraints and eq. (A18) to the equilibrium equation yields

$$W_b = 1.8 + K_{ub}U - 2.67\ I_{max}. \tag{A19}$$

The gain K_{ub} was arbitrarily set to 1.0, since the net inhibitory effect on the binocular layer could be controlled by parameters affecting the equilibrium level of the inhibitory cell U.

Eq. (6), the equation for the relay layers, reduces to

$$I = nW_i, \tag{A20}$$

where n is the number of discrete points of the binocular layer that remain excited in the equilibrium. Experience with the simulation system showed that an appropriate maximum value for n was 2. W_i was arbitrarily given the value 0.2. This value was chosen in order to keep the excitation level in the relay layer in the range from 0 to 1. With these values for W_i and n the maximum expected level of relay excitation I_{max} is 0.4.

The equilibrium equation for the inhibitory cell U is

$$U = nK_{bu}\Delta q. \tag{A20}$$

In the simulation the retinal-position increment Δq was 0.0375. The value 20 was chosen for K_{bu} giving U the value 1.5 when 2 points remain excited in the binocular layer.

Applying the various parameter choices to eqs. (A18) and (A19) gives K_{tb} the value 0.27, and W_b the value 2.23.

Parameters were experimentally adjusted from these starting values in order to improve transient performance. The resulting nominal parameter settings are given in Table 4.

Table 4 - Prey Localization Model Simulation Parameter Settings.

Simulation Time Increment, $\Delta t = 0.025$

Time Constants

$T_a = 0.40$ $T_t = 0.05$ $T_b = 0.10$ $T_u = 0.05$

Saturation Functions

	h_0	h_1
f_t	0.00	1.00
f_b	0.05	1.05
g	0.00	—

Spread Functions

	w_0	w_1	s_1	s_2	W
w_t	27.0	12.0	0.015	0.030	0.75
			(1.35°)	(2.70°)	
w_b	89.0	38.5	0.013	0.030	2.50
			(1.17°)	(2.70°)	
w_i	4.8	4.8	0.080	0.125	1.80
			(7.20°)	(11.25°)	

Gains

$K_{tb} = 0.25$ $K_{at} = 0.25$ $K_{bu} = 20.0$ $K_{ub} = 1.00$

APPENDIX B

SIMULATION OPTICS

In this appendix we describe the equations of the optical system used to provide input to both models. We describe the translation from spatial to retinal coordinates, outline the method used to simulate lenses and prisms in front of the eyes, and define the method used to calculate initial depth estimates from disparity and accommodation.

Optical Geometry

The optical equations are all based upon the configuration depicted in Fig. 49. Cartesian measurements are made relative to the x and y coordinates indicated by the bold orthogonal lines crossing in the center of the figure. The pupils of the two eyes are indicated by the bold dots near the bottom of the figure. The optical axis of each eye is indicated by the solid line passing through the pupil and labeled OA. The two optical axes intersect at the point $(0,y_f)$ indicated by the star. The retinas are indicated by the semicircles behind the pupils. The diagonals passing through the pupils are drawn orthogonal to the optical axes. Retinal angles, θ_L for the left eye and θ_R for the right eye, are measured relative to the optical axis, with sign as indicated at the ends of the retinal semicircles. The point (x,y) is an arbitrary spatial reference point, and the dashed lines are reference lines used in the derivations. The symbol l denotes the distance of the interpupillary line from the origin of the coordinate system, and w represents one-half of the interpupillary

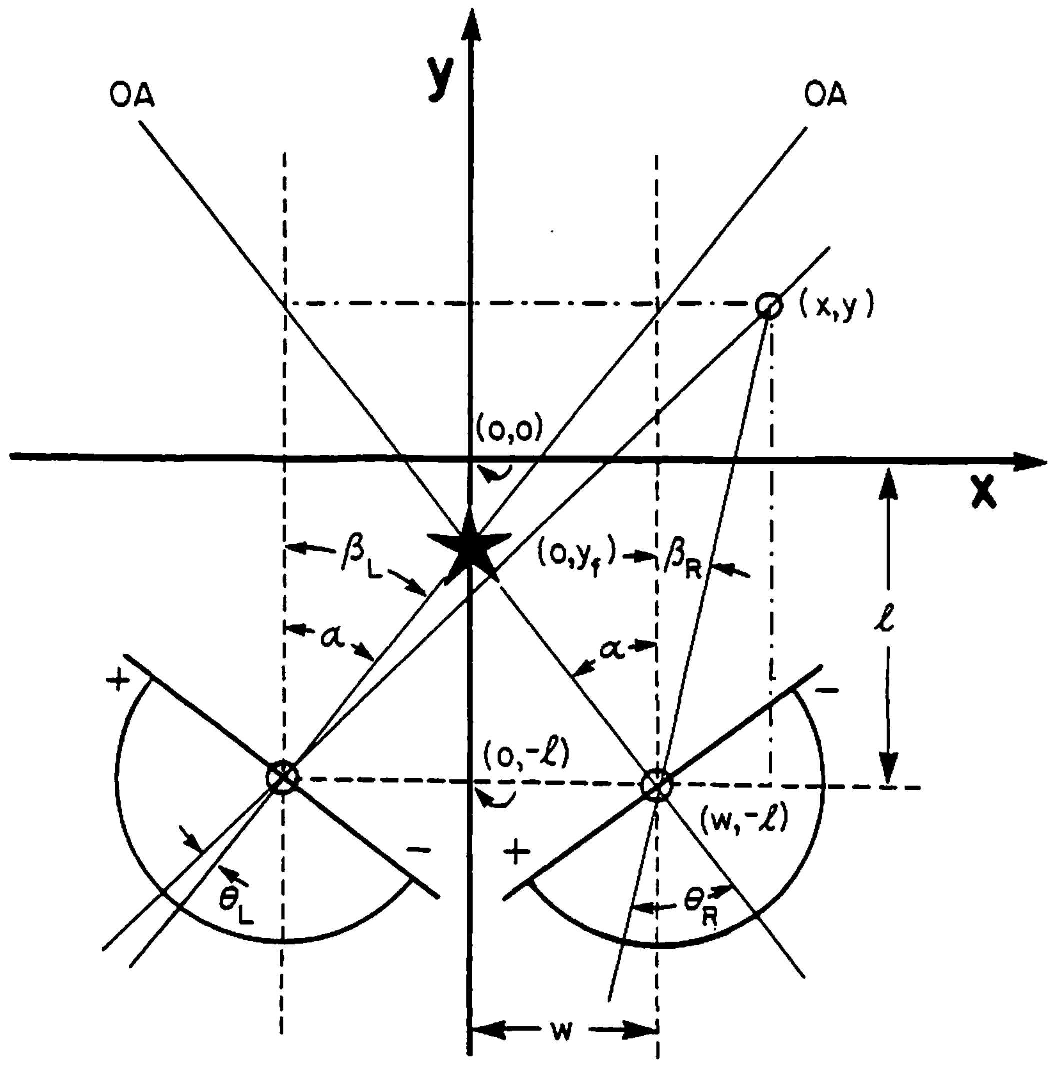


Fig. 49 - The Optical System.

distance. α is the angle of declension of the optical axis from the ordinate. Similarly, β_L and β_R are the angles of declension of the rays passing through point (x,y) and, respectively, the left and right pupils.

Although the eyes of frogs and toads are outward facing (their optical axes cross behind the eyes) we employed an optical model with eyes facing towards a frontally located fixation point. This was done to simplify the simulation system, and is consistent with the fact that binocular tectal projections have been shown to be in register with respect to a frontal horopter [Gaillard & Galand, 1980]. Further, the use of an unmoving fixation point has no effect upon the optical equations except to shift the center of the resulting retinal position versus disparity coordinate system; the fixation point in space maps to the origin of the internal coordinate system.

The retinal angles θ_L and θ_R of the projections of point (x,y) are given by

$$\theta_L = \beta_L - \alpha,$$

and

$$\theta_R = \beta_R + \alpha.$$

Inspection of Fig. 49 shows that

$$\alpha = \tan^{-1} \frac{w}{y_f + l},$$

$$\beta_L = \tan^{-1} \frac{x + w}{y + l},$$

and

$$\beta_R = \tan^{-1} \frac{x - w}{y + l}.$$

Finally, retinal angle is converted to retinal position by simply multiplying by a magnification factor. We chose retinal position 1.0 to represent 90°. Thus, if a is the magnification factor, q_L is position on the left retina, and q_R is position on the right retina we have

$$
\begin{aligned}
a &= 2 / \pi, \\
\alpha &= \tan^{-1} \frac{w}{y_f + l}, \\
q_L &= a \left(\tan^{-1} \frac{x + w}{y + l} - \alpha\right), \\
q_R &= a \left(\tan^{-1} \frac{x - w}{y + l} - \alpha\right).
\end{aligned}
\qquad \text{(B1)}
$$

Representation of Prisms

The presence of prisms in front of the eyes was simulated by applying an offset, proportional to prismatic strength, to the retinal coordinates determined by eqs. (B1). We were interested in the effect of prisms in altering the disparity (positional difference) between the projections of a point on the left and right retinas. Thus, we expressed the prismatic strength parameter K_p as a percentage of the maximum disparity range d_{max} which we wished to consider. The prismatic offset C_p and the augmented equations for retinal position are given by

$$
\begin{aligned}
C_p &= 1/2\, d_{max}(K_p / 100), \\
q_L &= a \left(\tan^{-1} \frac{x + w}{y + l} - \alpha\right) - C_p, \\
q_R &= a \left(\tan^{-1} \frac{x - w}{y + l} - \alpha\right) + C_p.
\end{aligned}
\qquad \text{(B2)}
$$

Retinal Projections

The projection of visual data onto the retinas was simplified by the fact that the simulation software used to experiment with the models records the dimensions and location of each object in the visual space. The arrays representing the retinas were loaded by using eqs. (B2) to determine the loci of the projections of each of these known objects onto the two retinas. Retinal points which received the projections of one or more objects were assigned the value 1. Points which received no projection were assigned the value 0. No attempt was made to account for occlusion of one object by another. Thus, if q is the retinal position coordinate, and R_L and R_R are the retinas

$$R_L(q) = \begin{cases} 0, & \text{no object projects to point q on left retina} \\ 1, & \text{any object projects to point q on left retina,} \end{cases}$$
$$R_R(q) = \begin{cases} 0, & \text{no object projects to point q on right retina} \\ 1, & \text{any object projects to point q on right retina.} \end{cases} \qquad \text{(B3)}$$

In the computer simulations the retinas were modeled as vectors of length 161.

Disparity Input Planes

Depth estimates based upon disparity were calculated from the retinal images and stored in planes organized according to a coordinate system based on retinal position q and disparity d. The binocular disparity inputs, D_L for the left side and D_R for the right side, were computed by the equations

$$D_L(q,d) = R_L(q) \cdot R_R(q+d),$$
$$D_R(q,d) = R_R(q) \cdot R_L(q-d), \tag{B4}$$

The range of allowable disparities d_{max} was constrained to represent a shift of one retina over the other of no more than 25% of the binocular field in either direction. In the computer simulation the disparity planes were modeled as 41 × 41 arrays.

Accommodation Input Planes

The accommodation input planes, A_L and A_R, were also represented in the retinal angle q versus disparity d coordinate system. Depth estimates based upon lens accommodation were approximated by building each accommodation input plane via a rule which for each stimulated point on the corresponding retina produced a set of values distributed along the disparity coordinate. A Gaussian distribution centered about the correct depth value was assumed. Thus, the value at a point (q,d) in one of the accommodation input planes could be considered to be either an estimate of depth or a measure of the quality of focus of the image at retinal position q with the lens accommodated to a depth corresponding with disparity d. If point a, in space, projects to points q_L on the left retina and q_R on the right retina then the disparity of point a is defined to be

$$d_a = q_R - q_L.$$

The full range of accommodative depth estimates for the point a is given by

$$A_L(q_L,d) = \exp[-(100\,\frac{d - d_a}{\sigma\, d_{max}})^2 \;/\; 2],$$

$$A_R(q_R,d) = \exp[-(100\,\frac{d - d_a}{\sigma\, d_{max}})^2 \;/\; 2], \tag{B5}$$

where: σ is the spread parameter measured in percent of maximum disparity d_{max}.

Representation of Lenses

A simulation of the presence of lenses in front of the eyes was provided by the introduction of a parameter K_l governing systematic error in the centering of the depth estimate curve about the true depth of a point. If this parameter is expressed in percent of maximum disparity then the adjusted center of the accommodative depth estimate curve is given by

$$d_a = q_R - q_L + d_{max}K_l \;/\; 100. \tag{B6}$$

Eqs. (B5) and (B6) together describe the method used to build the accommodation input planes. These planes were represented in the simulation as 41 $\times$ 41 arrays.

Conversion from Internal to External Coordinates

It was necessary in the simulations to be able to convert depth estimates expressed in the internal retinal angle versus disparity coordinate system back into the spatial coordinate system. From eqs. (B2) and inspection of Fig. 49 it can be demonstrated that given a disparity d and a retinal position q_L on the left retina or

q_R on the right retina a unique spatial point (x,y) is determined. Once a pair of retinal angles is determined from either of the pairs of formulae

$$\theta_R = q_R / a, \quad \theta_L = (q_R - d) / a,$$
$$\theta_L = q_L / a, \quad \theta_R = (q_R + d) / a, \tag{B7}$$

then the spatial coordinates are given by

$$y = 2\,w / [\tan(\theta_L + \alpha) - \tan(\theta_R - \alpha)] - l,$$
$$x = (y + l) \tan(\theta_R - \alpha) + w. \tag{B8}$$

Nominal Parameter Settings

The nominal parameter settings used in all of the reported experiments are given in Table 5.

Table 5 - Optical Simulation Parameter Settings.

y Coordinate of Fixation Point, y_f	$= -10$ cm.
Distance Between Pupils, $2w$	$= 6.0$ cm.
w	$= 3.0$ cm.
Distance of Interpupillary Line from Origin, l	$= 22$ cm.
Retinal Angle to Retinal Position Conversion, a	$= 2 / \pi$
Accommodation Spread, σ	$=$ 25% (cue interaction model)
	50% (prey localization model)
Prism and Lens Strength, K_p	$= K_l = 0$
Maximum Disparity, d_{max}	$= 0.25$

APPENDIX C

SOFTWARE SYSTEM FOR TESTING THE MODELS

The software system developed during this study was designed to be flexible enough to serve as a test-bed for experimenting with a variety of models. The system consists of an integrated collection of modules containing programs and subroutines that may be tailored by the experimenter to meet his or her particular requirements. It is implemented in VAX-Pascal on a VAX 11/780 running VMS.[1] The system can be run in either interactive or batch mode. In interactive mode the experimenter operates the system with a joystick, and has the capability to view the model's performance using a variety of displays. Interactive graphics are displayed in color on the monitor of a 512×512×16 Grinnell frame-buffer system. In batch mode the user prepares a "script file" that describes the various parameter settings and experiments to be performed during a run. In this mode the system can produce a graphics "Metafile" to be saved and examined later on the Grinnell display or output on a Symbolics laser printer for hardcopy.

This appendix is divided into two main sections. The first section is an operator's manual. It provides the instructions necessary to run simulation experiments once the system has been configured. The second section is a programmer's guide. It explains the steps required to configure the system to support a particular simulation.

[1] VAX and VMS are registered trademarks of the Digital Equipment Corporation.

Operator's Manual

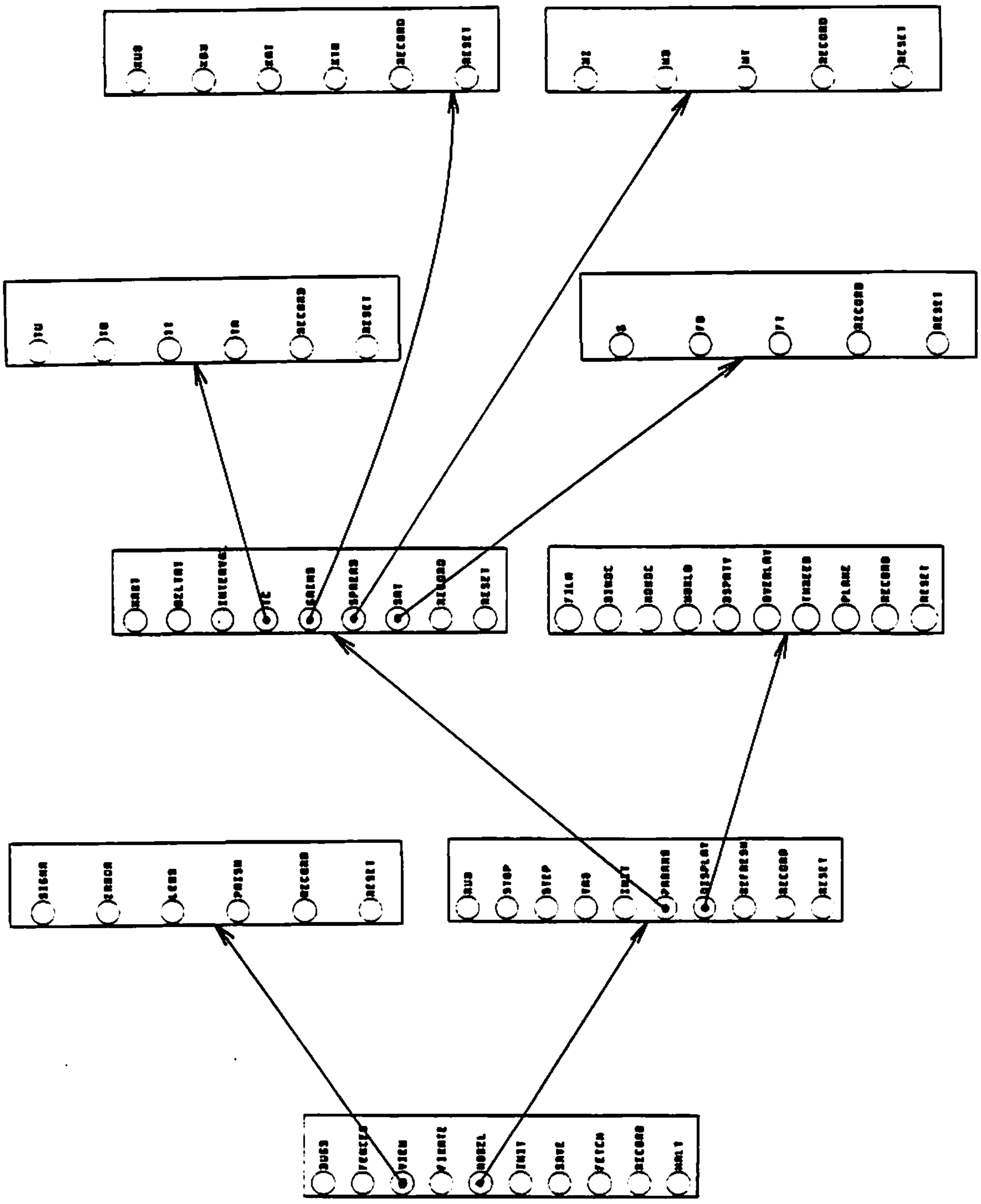
Whether in batch or interactive mode, the user accesses the system by selecting functions via a menu. In interactive mode the menu is displayed on the graphics screen, and selections are made using a joystick and pushbuttons. In batch mode the sequence of menu function selections is determined by a prewritten script. Initial sections of the Operator's Manual describe the layout of a typical menu, and the conventions used to access the menu in both interactive and batch modes. Later sections provide detailed descriptions of the various system functions that may be accessed via the menu.

The menu

The nodes of the menu are organized in a tree structure. Each leaf node on the tree represents a function that can be performed by the system. Each internal node on the tree has several children which, taken together, constitute a *group* from which an individual node can be selected. At any given time the user has access to one group. When an internal node is selected the user gains access to the group made up of its children. When a leaf node is selected the function associated with that node is executed.

The structure of the menu used to operate the simulation of the "prey localization model" is shown graphically in Fig. 50. It may be referred to when examining the examples used later in this manual. In this figure, the rectangles containing labeled circles represent groups of nodes. Arrows connect parent nodes with the group of nodes at the first level in the parent's subtree. Leaf nodes have

Fig. 50 - Menu Used with Prey Localization Model.


RULES
FENCES
VIEW
PIRATE
MODEL
INIT
SAVE
FETCH
RECORD
HALT
SIGMA
ERROR
LENS
PRISM
RECORD
RESET
RUN
STOP
STEP
VRS
INIT
PARAMS
DISPLAY
REFRESH
RECORD
RESET
START
DELTAT
INTERVAL
TE
GAINS
SPARES
SAT
RECORD
RESET
FILM
DISDC
MONDC
WORLD
DSPRTY
OVERLAY
THREED
PLANE
RECORD
RESET
TU
TB
TI
TA
RECORD
RESET
S
FB
FT
RECORD
RESET
XUS
XBU
XAT
XTH
RECORD
RESET
MI
MB
MT
RECORD
RESET

no arrows coming from them. The root node of the tree is not drawn but is, in effect, the operating system. When a simulation session is first initiated the group of nodes at the top of the menu tree is activated. The user accesses a particular function by moving through the tree to the function's group and then selecting the function. For example, to access the PRISM function the user would first select VIEW, giving him access to the group containing the PRISM, LENS, ERROR, and SIGMA functions, and then select PRISM.

Interactive mode overview

The primary use of the interactive mode is to allow the experimenter to tune model parameters and obtain a feel for the model's performance. Parameters can be readily adjusted, experiments can be configured and run, and model performance can be observed with a variety of displays. An interactive session is initiated by using the VMS command RUN to activate the executable image of the interactive system.

Once the system is activated the user interacts with it by using the Grinnell's control box. The control box has a joystick that is used to control the position of the cursor on the display screen, as well as several pushbuttons and switches. Before attempting to operate the system the user should:

1) place the cursor in "tracking" mode, by raising the TRACK switch;

2) enable cursor number one by raising the switch numbered 1;

3) disable cursors 2 through 4 by lowering switches 2, 3, and 4.

The two pushbuttons monitored by the system are labeled HOME and ENTER. ENTER is used to select a function or initiate an action. The user generally leaves a function by depressing HOME immediately followed by ENTER. This sequence is denoted by HOME-ENTER in the following text.

In interactive mode, the currently active group of menu nodes is displayed in a grey rectangle positioned along the right-hand edge of the graphics screen. Each node is represented by a labeled black disk. The user selects a node by moving the cursor onto the appropriate menu disk and then depressing ENTER. The system indicates the selection by changing the color of the selected disk, and its associated text, to yellow. If the selected node is an internal node the system responds by descending one level in the menu tree and displaying the group of nodes containing the selected node's children. If the selected node is a leaf node then its corresponding function is executed. The user ascends to the parent node of the active group by either depressing HOME-ENTER or selecting the RESET node at the bottom of the menu display. Depressing HOME-ENTER or selecting HALT while at the top level of the menu tree terminates the interactive session by returning control back to the operating system.

By far the most commonly used menu function is one to alter the value of a scalar parameter. When the user selects a menu node associated with a scalar parameter, a colored bar will appear near the center of the display along with text giving the name and the current value of the parameter. A blue bar indicates that the parameter is positive and a red bar indicates that it is negative. The cursor will appear centered on the top edge of the bar. Moving the cursor down will decrease the value of the parameter, and moving it up will increase its value. Depressing HOME-ENTER will return control to the menu. There is no provision for numeric parameters to be entered via the keyboard in interactive mode.

In every menu group there is a node labeled RECORD. Selecting this node will cause an image of the display screen to be stored in a disk file named PLOT.PLT. A new version of PLOT.PLT is created every time RECORD is selected. This file is in an appropriate form for later printing using the graphics mode of the Printronix printer. Scanning the display's framebuffer to build this file takes about one minute. The progress of the scanning operation is indicated by the position of the cursor on the display screen. A more sophisticated scheme for obtaining hardcopy is built into the batch version of the system.

Batch mode overview

Batch mode allows the user to perform a series of experiments without manual intervention. It is most useful once a reasonable set of operating parameters has been determined. Batch mode is the preferred means of operating the system when the experimenter wishes to run a more methodical series of experiments than is possible under interactive control. Other advantages of batch mode are that numeric values are explicitly given so that they are more tightly controlled, and a "metafile" capability is provided to allow recording of graphics for later conversion to hardcopy. The batch system is activated from a batch command stream. This command stream is prepared ahead of time and then initiated using the VMS command SUBMIT.

The batch system consists of two executable images. The first of these is called BATCHSYS. It is a language translator that converts a script, written in a simple Script Description Language (SDL), into a form that is usable by the simulation system. The second executable is the batch version of the simulation system. It operates much like the interactive system but instead of obtaining its commands from the joystick it uses the object form of the script to determine its sequence of actions. Fig. 51 is a block diagram showing the inputs and outputs from these two programs.

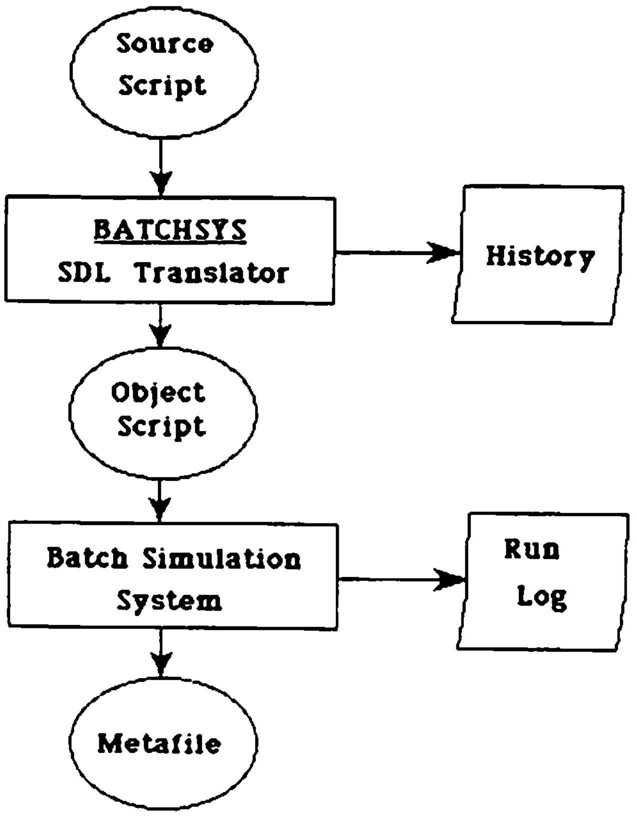


Fig. 51 - Batch System Block Diagram.

BATCHSYS takes a source language version of the script as its only input. As output it produces a History File and an Object File. The History File includes a listing of the source script, with any translation-time errors noted, and a readable version of the object script. It may be printed for off-line analysis. The Object File contains the object version of the input script, and is the only input to the Simulation System. The Simulation System produces a Run Log File and a Metafile. The Run Log contains a summary of the simulation run and indicates any run-time errors detected in the script. The Run Log may be printed for off-line analysis. Graphics images may be saved during a run using the SDL command RECORD. All saved images are written to the Metafile. The Metafile can be processed by the Metafile Display Program of the GUS system [Cromarty & Sutton, 1983] for previewing on the Grinnell display or to produce hardcopy on the Symbolics laser printer. A sample command stream for running the batch system is shown in Fig. 52.

```
$
$ set default [neudhh.frog.lens]  ! select default directory
$
$ define scriptfile script.dat    ! script.dat contains SDL script
$ define objectfile script.obj    ! write object script to script.obj
$ define historyfile sys$output   ! write history to batch log file
$ run batchsys                    ! run the SDL translator, BATCHSYS
$
$ define object script.obj        ! script.obj contains object script
$ define metafile images.mfi      ! write metafile to images.mfi
$ define runlog sys$output        ! write run log to batch log file
$ run batchlens                   ! run the batch simulation system
metafile                          ! write graphics to metafile
$
$ delete script.obj;*             ! discard object file when done
$
```

Fig. 52 - Sample Batch Command Stream.

The command stream of Fig. 52 starts with a "set default" command to establish the directory containing the simulation system as the default directory. The program BATCHSYS refers to the Script, Object, and History files by the symbolic labels "scriptfile," "objectfile," and "historyfile." The "define" commands preceding the command "run batchsys" bind actual file names to these labels. The batch version of the simulation system refers to the Object, Meta-, and Run Log Files by the symbolic labels "object," "metafile," and "runlog." The three "define" commands preceding the command "run batchlens" bind actual file names to these labels. Assigning the symbolic labels "historyfile" and "runlog" to "sys$output" causes the History and Run Log Files to be written to the batch log file so that they will be included in the printed summary of the batch run. The program name "batchlens" is only representative; the actual name of the batch version of the simulation system is chosen by the user when configuring the system. The line following the "run batchlens" command contains a mnemonic indicating the device to which graphics will be written. The mnemonic "metafile" informs the batch system that all graphics are to be written to the Metafile. The mnemonic "null" may be used in place of "metafile" to indicate that no graphics are to be produced. Other options are available (see the GUS Manual [Cromarty & Sutton, 1983]) but are generally not useful. The final "delete" command, to discard the Object file at the end of the run, is optional.

Syntax diagrams for the Script Description Language (SDL) are provided in Fig. 53. In each diagram, rectangles represent syntactic units described in other diagrams, circles and rounded rectangles represent input characters, and the lines

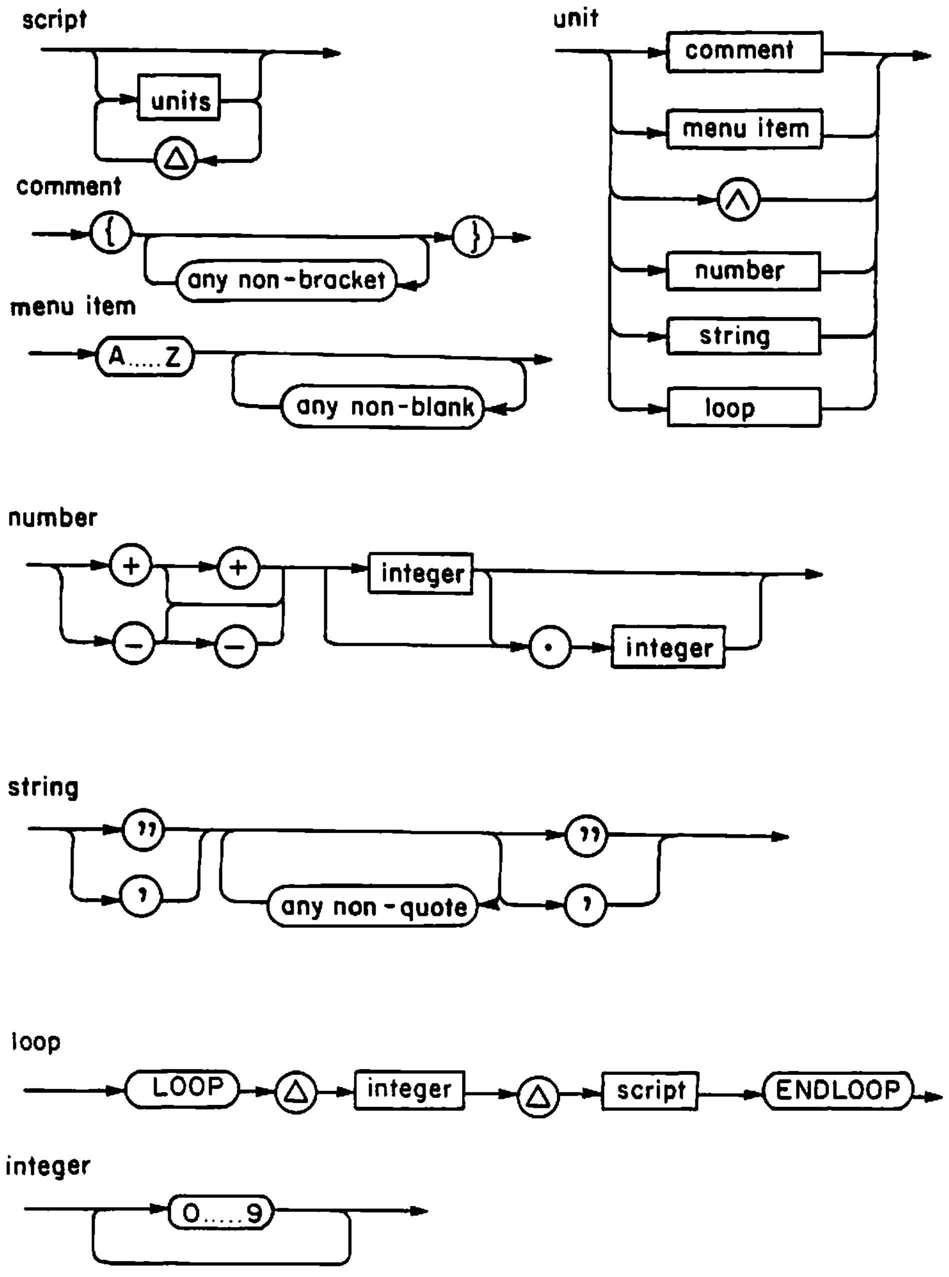


Fig. 53 - SDL Syntax Diagrams.

with arrows trace syntactically-correct paths through a diagram. The symbol Δ represents "white-space," that may consist of one or more space, end-of-line, or tab characters. Comments are contained within curly brackets. They can be freely interspersed with code but may not be nested. All alphabetic characters are converted to upper-case before being processed.

The only control construct in SDL is the LOOP statement. There is no provision for conditional execution. The LOOP statement takes an unsigned integer constant as its only argument. This argument determines the number of times that the loop will iterate. The end of the block of statements contained within a loop is delineated by an ENDLOOP statement. Loops may be nested to any level.

The two kinds of statement controlling position in the menu tree are 1) menu node keywords, and 2) the $\wedge$ command. Every node in the menu has a label associated with it; in interactive mode this label is displayed to the right of the circle representing the node in the menu display, as shown in Fig. 50. In SDL this same label is used as a keyword for the corresponding menu node. Menu item keywords are limited to 8 characters. The presence of a keyword in the script causes the corresponding menu node to be selected, if it is in the currently active group. The check, to determine whether or not a particular keyword corresponds with a node in the currently active group, is made at translation time. The $\wedge$ command causes control to ascend to the group containing the parent node of the currently selected menu group. It is equivalent to depressing HOME-ENTER or selecting RESET (or HALT) in interactive mode. The RESET and HALT menu nodes are not accessible in batch mode. A script is normally terminated by a sequence of enough ∧ commands to return control to the root node (the operating system). Missing ∧ commands will be automatically appended to the end of a script.

A function, activated by a keyword selecting the corresponding menu node, may take either real-valued numerics or character strings as arguments. Arguments are placed in the script following the node selection keyword and in the order in which they will be accessed by the associated function. Numerics are interpreted by the functions as either replacement or incremental values. If the number is preceded by either no sign or a single + or − sign it is taken as a *replacement* value. ++ or −− before a number indicates that it is an *incremental* value. Character string arguments are contained within either single (') or double (") quote marks. There is no provision for including quotes within a character string, but all other ASCII characters can be used. Character strings are truncated on the right to a maximum of 80 characters.

Many of the menu nodes are associated with a function that alters the value of a scalar simulation parameter. In batch mode this function requires a single numeric argument. The argument is used to replace or increment the current value of the parameter. An argument preceded by ++ will be added to the parameter's current value. One preceded by −− will be subtracted from the current value. A simple signed or unsigned argument will replace the parameter's current value. A record of the updated value of the parameter will be written to the Run Log.

The batch simulation system maintains an internal virtual graphics display that, for any given function, mimics the display provided by the interactive system. Every menu group contains a node labeled RECORD. Selecting this node while in batch mode causes an image of the graphics display to be recorded in the Metafile. The RECORD function takes a single, optional, character string parameter. If present, this parameter is written to the Metafile ahead of the graphics display, and serves as a label for the image. Only one Metafile is produced by a single batch run. Images are logically separated within this file by control records. After a batch run, the GUS Metafile Display Program can be used to either preview the Metafile's contents on the Grinnell monitor or print its contents on the Symbolics laser printer.

Fig. 54 shows a sample script written in SDL to pre-set the values of all parameters used in the simulation of the "prey localization model." The "save" command, at the end of the script, saves the updated parameters in the file SAVELENS.SAV. Although some of the functions shown have not yet been described, this figure can be used in conjunction with Fig. 50 as an example of how a script may be used to traverse a menu tree and access its functions.

The scene construction functions

The scene construction functions on the menu allow the user to position prey (bug) and barrier (fence) objects in the input scene, to adjust parameters affecting how this scene will be projected onto the simulated retinas, and to save and restore parameters using a disk file.

```
{Build a base set of parameters for prey localization model}

view
  prism 0.0 lens 0.0 error 0.0 sigma 50.0 ∧

fixate 0.0

bugs "clear"
fences "clear"
record "empty arena"

model
  display
    overlay world monoc ∧

  ya0 0.0

  params
    deltat 0.025 interval 0.1 maxt 5.0
    tc
      ta 0.4 tt 0.05 tb 0.1 tu 0.05 ∧
    sat
      ft 0.0 1.0
      fb 0.05 1.05
      g 0.0 1.0 ∧
    spread
      wt 27.0 12.0 0.015 0.03
      wb 89.0 38.5 0.013 0.030
      wi 4.8 4.8 0.08 0.125 ∧
    gains
      ktb 0.25 kat 0.25 kbu 20.0 kub 1.0 ∧ ∧ ∧

save "savelens.sav"
∧
```

Fig. 54 - Sample SDL Script.

Placement of bugs and fences. When the system is first initiated, control is at the top level of the menu. At this level the graphics display depicts a 40 cm. × 40 cm. arena. A scene may be constructed within this arena by using the BUGS and FENCES functions to place bugs and fences at desired locations.

The operation of these functions in batch mode is different enough from their operation in interactive mode that they are explained separately below.

When the BUGS function is selected in interactive mode the cursor appears in the center of the arena. A bug is added to the scene by positioning the cursor in the arena and depressing ENTER. The bug is represented by a red rectangle. The lower left-hand corner of this rectangle is placed at the cursor position. A bug is removed from the scene by positioning the cursor on the bug and depressing ENTER. To return to the menu either depress HOME-ENTER or move the cursor outside of the arena and depress ENTER.

In batch mode the BUGS function takes an arbitrary number of pairs of numeric arguments that may, optionally, be preceded by the character string argument "CLEAR." The arguments are processed in the order in which they appear. The "CLEAR" argument causes all bugs to be removed from the scene. Numeric arguments refer, respectively, to the x and y coordinates (measured in centimeters from the center of the arena) of the lower left-hand corner of the rectangle representing a bug. The numerics in a pair of arguments must both have either replacement or incremental values. It is not legal to have one replacement value and one incremental value in the same argument pair. A pair of replacement numerics can be used to either add a bug to the scene or remove a bug from the scene. Replacement arguments specify (x,y) coordinates. If these coordinates fall on an area of the arena currently occupied by a bug that bug will be removed. If these coordinates do not fall on an existing bug, a new bug will be added to the scene. As bugs are created their coordinates are added to the *beginning* of a list.

Incremental arguments are used to change the position of bugs that are currently on the list. The first pair of incremental arguments adjusts the coordinates of the first bug on the list (the most recently created bug), the second incremental pair adjusts the coordinates of the second bug on the list, etc. For example, in SDL the sequence

```
BUGS "CLEAR" 10.0 5.0 −10.0 5.0
```

first removes all bugs from the scene and then adds two new bugs. After execution, the first bug on the list (the second bug created) will have arena coordinates (−10, 5) and the second bug on the list (the first bug created) will have coordinates (10, 5). If, later, the SDL sequence

```
BUGS ++5.0 −−2.5 −−5.0 −−2.5
```

is executed the coordinates of the first bug on the list will be changed to (−5, 2.5) and those of the second bug will be changed to (5, 2.5). This scheme is a bit confusing so the user is cautioned to exercise care in use of the BUGS function.

When the FENCES function is selected in interactive mode the cursor appears in the center of the arena. To add a fence to the scene position the cursor where one end of the fence should be placed and depress ENTER. A magenta dot, indicating the selection, will appear at the cursor position. Then position the cursor where the other end of the fence should be placed and again depress ENTER. A series of green fenceposts will appear connecting the two selected end points. An existing fence can be deleted, translated, stretched, or rotated. To delete a fence, place the cursor on the left-most fencepost and depress ENTER. To translate a fence, place the cursor on any internal fencepost and depress ENTER. The color of the fenceposts will change to magenta, indicating that the fence is to be modified. Move the cursor to the desired new position of the selected post and again depress ENTER. The entire fence will be moved to the new position and displayed again in green. The translation will be uniform and without rotation. To rotate or stretch a fence, place the cursor on the right-most fencepost and depress ENTER. The color of the fenceposts will change to magenta. Move the cursor to the desired new position of the selected fencepost and depress ENTER. The fence will be stretched and/or rotated about its left-most post and the modified fence will appear in green. The number of posts in the modified fence will be the same as in the original fence, so stretching a fence will increase the distance between posts and shrinking it will decrease this distance. To return to the menu either depress HOME-ENTER or move the cursor outside of the arena and depress ENTER.

In batch mode the FENCES function takes an arbitrary number of pairs of numeric arguments that may, optionally, be preceded by the character string argument "CLEAR." The arguments are processed in the order in which they appear. The "CLEAR" argument causes all fences to be removed from the scene. Numeric arguments refer, respectively, to the x and y coordinates (measured in centimeters from the center of the arena) of a fencepost. The numerics in a pair of arguments must both have either replacement or incremental values. It is not legal to have one replacement value and one incremental value in the same argument pair. Replacement numerics can be used to either add a fence to the scene or remove a fence from the scene. Replacement arguments specify (x,y) coordinates. If these coordinates fall on any fencepost of an existing fence that fence is removed. If these coordinates do not fall on an existing fencepost, a new fence is added to the scene. In this case a second pair of replacement arguments must immediately follow the first pair. The two pairs, taken together, specify the positions of posts at the two ends of the fence. As a fence is created its description is added to the *beginning* of a list. Incremental arguments are used to change the positions of the two ends of a fence. A first pair of incremental arguments adjusts the position of the left-hand end of the first fence on the list. These arguments must be followed by a second pair of incremental arguments to adjust the position of the right-hand end of the first fence on the list. A second group of four incremental arguments will adjust the position of the second fence on the list, etc. For example, in SDL the sequence

```
FENCES "CLEAR" −10.0 5.0 10.0 5.0 −5.0 −5.0 5.0 −5.0
```

first removes all fences from the scene and then adds two new fences. After execution, the left end of the first fence on the list (the second fence created) will have arena coordinates (−5, −5) and its right end will have coordinates (5, −5). The second fence on the list (the first fence created) will be drawn between the points (−10, 5) and (10, 5). If, later, the SDL sequence

```
FENCES ++0 ++5 ++0 ++5 −−5 ++0 ++5 ++0
```

is executed the coordinates of the first fence on the list will be changed to [(−5, 0), (5, 0)] and the coordinates of the second fence on the list will be changed to [(−15, 5), (15, 5)]. This scheme is a bit confusing so the user is cautioned to exercise care in use of the FENCES function.

Changing viewing parameters. The viewing parameters are those parameters that affect how the scene in the arena will be projected onto simulated retinas. The reader wishing a detailed description of these parameters should refer to Appendix B.

The fixation point determines which point in the arena maps to the origin of the retinal angle vs. disparity coordinate system. The fixation point is always aligned on the body axis of the simulated animal. Thus, only its depth coordinate y_f can be varied. To adjust the fixation point select FIXATE on the menu. In interactive mode the cursor will be placed at the current fixation point. Move the cursor to the desired new location and depress ENTER. The fixation point will be moved to the new position and the cursor will return to the menu. In batch mode a single numeric argument should follow the FIXATE command in the script. Depending on the form of this argument, it will be used to replace or increment the current value of y_f.

Selecting VIEW on the menu provides access to four scalar viewing parameters: PRISM, LENS, SIGMA, and ERROR. PRISM is used to adjust the prism strength parameter K_p, and LENS to adjust the lens strength parameter K_l. The spread parameter σ is used in constructing the accommodation input planes. It is adjusted via SIGMA. Although this feature was not used for any of the simulations reported in this dissertation, the system also supports the inclusion of a random error in the calculation of the accommodation input planes. The parameter controlling this error is accessed via ERROR. The value of this parameter determines the width of the density function of a uniformly distributed random variable that is added to the lens parameter K_l immediately prior to constructing the accommodation input planes.

Saving and restoring system parameters. The values of all adjustable parameters (both viewing parameters and simulation control parameters – see next section), and the positions of all bugs and fences in the arena, may be saved in a file for later retrieval. Selecting SAVE will cause these values to be saved and selecting FETCH will cause them to be retrieved. In interactive mode the system will prompt the user for a file name after either SAVE or FETCH is selected. Using the keyboard, the user may either enter a file name or depress return without entering a name. In batch mode a single character string parameter may be used to provide a file name. In either mode, if a file name is not provided then the most recently entered file name is used, or if no name has been entered during the run a preprogrammed default is used.

The system parameters can be restored to preprogrammed values by selecting INIT at the top menu level. This function is much less flexible than the SAVE/FETCH functions, and is only occassionally useful in practice.

The simulation control functions

The simulation control functions allow the user to adjust simulation parameters, run the simulation, and select and view the various displays of the simulation's state. Access to these functions is gained by selecting MODEL while at the top level of the menu. Parameters adjusted by the simulation control functions may be saved by returning to the top menu level and selecting SAVE as described in the previous section. When MODEL is selected, the system initializes the simulation by 1) calling the optical simulation procedures to project the current scene onto modeled retinas and build the accommodation and disparity input planes, and 2) setting simulation variables to their initial values. After this is completed, the menu group descending from the MODEL node is activated.

Controlling the displays. After MODEL is selected, the currently active display of the model's state may be drawn on the graphics screen by selecting REFRESH. Various functions cause this display to be overwritten. The display can be restored by again selecting REFRESH.

There are several types of display which can be used to depict the model's current state. Selecting the DISPLAY function gives the user access to four switches which control the display type. One menu node is allocated to each of the possible states of these switches. The current state of each switch is indicated by a large yellow arrow pointing to the menu node representing that state.

Menu nodes PLANE, THREED, and OVERLAY represent the three states of a switch that determines the method to be used to display two dimensional fields. If PLANE is selected, a two dimensional field will be displayed as a grid. The potential in the region of the field covered by a particular grid element is indicated by the light intensity of that element. If THREED is selected, potential in the field will be indicated by a contour plot. OVERLAY is used to display the potential of a field represented in the internal (retinal angle vs. disparity) coordinate system but mapped back onto the external (x,y) coordinate system and overlaid upon an image of the scene in the arena. The display may be thought of as being divided into a grid. At each grid position a small oval will be drawn whose size indicates the potential in the field. If the potential is below a small preprogrammed threshold no oval is drawn.

Menu nodes DSPRTY and WORLD represent the two states of a switch that determines which coordinate system will be used to display internal two-dimensional fields. If DSPRTY is selected, fields will be displayed in the internal (retinal angle vs. disparity) coordinate system. If WORLD is selected, internal fields will be mapped back onto the external (x,y) system before being displayed. If the simulation has not been run since the last time it was initialized, the displays will show input planes. If the simulation has been run, the displays will show planes representing the current state of the simulation. OVERLAY mode only works if WORLD has been selected.

Menu nodes MONOC and BINOC represent the two states of a switch that determines whether displays in WORLD mode should be calculated from the model's accommodative or disparity planes. In MONOC mode, the accommodative planes are used. In BINOC mode, the disparity planes are used.

Menu node FILM toggles an on/off switch used to put the system in a special mode for making animated films. If FILM is selected, the state of the switch reverses. This mode is only available in interactive mode. Filming has only been attempted once and, due to mechanical problems, was not very successful. Further documentation on producing films from the system is provided in [Overton, 1980] and in the source code for the system.

Selecting simulation parameters. Access to simulation parameters is gained by selecting PARAMS. The menu group activated by PARAMS contains nodes that allow the user to modify both simulation control parameters and model parameters.

The three nodes INTERVAL, DELTAT, and MAXT allow the user to modify scalar paramters which control the running of the simulation. DELTAT is used to set the simulation step size, INTERVAL is used to set the simulation display interval, and MAXT is used to set an upper bound on the simulation running time. For instance, if the step size is set to 0.1, the display interval to 0.5, and the upper bound to 2.0 then during a run of the simulation each time step (iteration of the simulation equations) will represent 0.1 time-units, the graphics display will be updated every 5 time steps (0.5 time-units), and the run will terminate after 20 time steps (2.0 time-units).

The node GAINS gives the user access to a group of nodes corresponding to the model's gain parameters. The node TC gives access to a group of nodes corresponding to the model's time constants. Gains and time constants are scalar parameters. The particular gain or time constant associated with a node is indicated by the node's label (refer to Fig. 50 for an example).

The node SAT gives the user access to a group of nodes corresponding to the model's saturation functions. The particular saturation function associated with a node is indicated by the node's label (refer to Fig. 50 for an example). To change the thresholds in a saturation function, select the appropriate node. An appropriately labeled graph of the function will be displayed. In interactive mode, the cursor will appear at the position on the horizontal axis corresponding to the function's threshold. The threshold may be adjusted by moving the cursor left or right. If ENTER is depressed the cursor will jump to the position corresponding to the function's saturation level. The saturation level may also be adjusted by moving the cursor. Depressing ENTER again will shift the cursor back to the position of the threshold. To return to the menu, depress HOME-ENTER. If the desired value of either the threshold or the saturation level is beyond the scale of the display, simply adjust its value as near as possible to the desired value, depress HOME-ENTER to return to the menu, and select the same saturation function again. This will cause the graph to be rescaled so that further adjustment is possible. In batch mode a saturation function is adjusted by specifying the keyword associated with its menu node, followed by one or two numeric parameters. The first numeric parameter is used to adjust the function's threshold, and the second (if present) is used to adjust the function's saturation level. Each parameter may have either a replacement or incremental value. A replacement value will replace the current value of the associated parameter, and an incremental value will be added to the parameter's current value. The same function and display are used to adjust saturation-threshold functions as are used to adjust simple threshold functions (refer to Appendix A, Fig. 47). If the function is a simple threshold function only the threshold parameter is used in the simulation calculations..

The node SPREAD gives the user access to a group of nodes corresponding to the model's spread functions. The particular spread function associated with a node is indicated by the node's label (refer to Fig. 50 for an example). A spread function is described by a piece-wise polynomial drawn through three knot points.

Details are provided in Appendix A (see especially Eq. (A6) and Fig. 48). To change a spread function's knot points, select the menu node associated with the spread function. An appropriately labeled graph of the function will be displayed. In interactive mode, the cursor will appear at the position on the vertical axis corresponding with the vertical coordinate of the first knot point, i.e. the height of the center of the spread function. The vertical position of this knot may be adjusted by moving the cursor up or down. If ENTER is depressed the cursor will jump to the position corresponding with the function's second knot point. This knot represents an intermediate position on the function's graph. Both its vertical and horizontal coordinates may be adjusted by moving the cursor. Depressing ENTER again will move the cursor to the third knot point, the point at which the spread function attains the constant value zero. The horizontal coordinate of this knot point may be adjusted by moving the cursor left or right. Depressing ENTER again will shift the cursor back to the position of the first knot point. To return to the menu, depress HOME-ENTER. If the desired position of any of the knot points is off the scale of the display, simply adjust its value as near as possible to the desired value, depress HOME-ENTER to return to the menu, and select the same saturation function again. This will cause the graph to be rescaled so that further adjustment is possible. In batch mode a spread function is adjusted by specifying the keyword associated with its menu node, followed by from one to four numeric parameters. The first numeric parameter will be used to adjust the vertical coordinate of the first knot point, the second will adjust the vertical position of the second knot point, the third will adjust the horizontal position of the second knot point, and the fourth will adjust the horizontal position of the third knot point. Each parameter may have either a replacement or incremental value. A replacement value will replace the current value of the associated knot coordinate, and an incremental value will be added to the coordinate's current value.

Running the simulation. The menu nodes STEP, RUN, and STOP are used to initiate and terminate runs of the simulation algorithm. Selecting STEP causes the simulation to run for as many time steps as are required to reach the next display interval (see DELTAT and INTERVAL above). Selecting RUN causes the simulation to run for as many time steps as are required to reach the maximum simulation time (see MAXT above). While either STEP or RUN is in effect, the graphics display will be updated after every display interval. In interactive mode RUN can be prematurely halted before the maximum simulation time is reached. When RUN is activated the cursor is placed in the center of the circle representing the menu node STOP. Depressing either ENTER or HOME-ENTER while in this state will cause the simulation to terminate at the next display interval. STOP is not available in batch mode. In interactive mode the current simulation time is displayed in a box labeled CLOCK just below the menu display. In batch mode, simulation time is output at the top of the display.

A final example

A final example of how the system is used to run a simulation is provided by the SDL script shown in Fig. 55. This script is an abbreviated version of the script used to test the "prey localization model" for a standard set of symmetrical bug placements and two different settings of the binocular layer spread function W_b. The various steps in the script are annotated with comments to provide a more detailed explanation.

```
        {Run to try various 2-bug configurations with two binoc. spreads}

fetch "savelens.sav"          {get saved parameters and scene}
fences "clear" bugs "clear"   {make sure scene is empty}

model                         {prepare to adjust parameters}
  ya0 0.0                     {neutral accommodation position}
  display
    world overlay ∧           {use overlay display mode}
  params
    maxt 2.0                  {stop simulation when time=2.0}
    spread                    {initial spread function Wb}
      wb 1.0 1.0 0.013 0.030 ∧ ∧ ∧

bugs −16 −15 14 −15           {initial bug placements at (14,−15) and (−16,−15)}
loop 7                        {use 7 different bug y coordinates}
  loop 3                      {use 3 different bug x coordinates}
    model                     {initialize simulation}
        {run the simulation and save final state in metafile}
      run record "Simulation Results, 1st Group" ∧
    bugs −−5 ++0 ++5 ++0 {move each bug 5 cm. nearer to midline}
  endloop
  bugs ++15 ++5 −−15 ++5 {restore bug x coords., add 5 cm. to y coords.}
endloop

model                         {repeat above with new Wb params.}
  params
    spread
      wb 29.7 12.8 0.013 0.030 ∧ ∧ ∧
bugs "clear" −16 −15 14 −15
loop 7
  loop 3
    model
      run record "Simulation Results, 2nd Group" ∧
    bugs −−5 ++0 ++5 ++0
  endloop
  bugs ++15 ++5 −−15 ++5
endloop
∧                             {terminate run}
```

Fig. 55 - SDL Script for Prey Localization Model Experiment.

Programmer's Guide

The simulation system was developed to support the modeling efforts outlined in this dissertation, and was not engineered as a general modeling package for distribution to users with little programming background. Instead, the system is a collection of Pascal programs and modules which can be configured by a knowledgable programmer to support a variety of simulations. This Programmer's Guide is intended to be used as a supplement to the system listings.[2] It will provide the programmer with an understanding of the layout of the system and the steps that must be taken in order to tailor the system's programs, data structures, and procedures to suit a particular application. This guide, together with the Operator's Manual and the listings, completes the documentation of the system.

The simulation system makes use of several graphics packages available at the Research Computer Facility of the Computer and Information Science Department at the University of Massachusetts. This Programmer's Guide does not provide documentation for these packages. Instead, the reader is referred to the documentation indicated by the references below. In interactive mode the Grinnell Support Package [Morse, 1979] provides the graphics primitives that enable the drawing of vectors, circles, rectangles, etc. In batch mode the Grinnell Emulator Package of the GUS [Cromarty & Sutton, 1983] device independent graphics system is used to provide these primitives. GUS also provides the "metafile" capability for saving graphics in a disk file while in batch mode. In both interactive and batch

[2] A copy of the system can be obtained from the author upon request.

modes, the DIGR [Morse, 1981] system is used to produce the higher-level displays that depict the simulation's state.

The software available with the system provides for:

1) menu driven control in both interactive and batch modes;
2) run-time construction of a two-dimensional scene consisting of prey (bug) and barrier (fence) objects;
3) projection of this scene onto two-dimensional retinas, with appropriate corrections for simulated lenses and prisms;
4) setting, saving, and restoring simulation parameters;
5) building graphics displays to depict the simulation's state;
6) running the simulation. A separate language-translator program translates scripts written in Simulation Description Language[3] (SDL) to a form usable by the simulation system when operating in batch mode.

In order to tailor the system to an application, the programmer must provide:

1) a description of the menu to be used by the application, and a set of menu control procedures;
2) descriptions of the various graphic displays, and a set of display procedures;
3) simulation and simulation-initialization procedures;
4) descriptions of simulation variables;
5) descriptions of simulation parameters, and their initial values;
6) settings for various predefined constants;

[3] A complete description of the syntax and semantics of SDL is contained in the Operator's Manual.

7) descriptions of application-specific data structures;

8) application-specific procedures.

Software overview

The software for the simulation system has evolved over a period of several years, and as such is scattered over many different source files, object libraries, and command files. All original software is in VAX Pascal, but many routines were "borrowed" from others and are written in FORTRAN. Three different versions of the system have been successfully configured. A version supporting the "cue interaction model" and another supporting the "prey localization model" were used to produce the results reported in this dissertation. The third version supports two models of detour behavior. File names reported in this document are those used with the version supporting the "prey localization model." File names beginning or ending with "LENS" are unique to this version; other versions have files with similar names but a different prefix or suffix. Most file names that do not begin or end with "LENS" are identical across versions.

The various Pascal source files which make up the main portion of the simulation system are listed in Table 6. The three *include*-files LENSDEFS.PIF, BATCHDEFS.PIF, and LENSPROC.PIF contain the definitions of all global constants, data types, variables, procedures, and functions used by the main simulation system. These files are *included* when the main simulation program in LENS.PAS is compiled. Therefore, these definitions become part of the environment file LENS.PEN which is shared by all of the simulation modules. The *include* file LENSCTINI.PIF contains the Pascal code for the procedures and functions which build and define the menu at run time. The program in BATCHSYS.PAS is the translator which converts an SDL script into object form. It gains access to a description of the system menu by *including* the files LENSDEFS.PIF and LENSCTINI.PIF. The environment files UTILITY.PEN, ENCODE.PEN, FASTCRSR.PEN, PLOT.PEN, DIGR.PEN, and GRINNELL.PEN define various interface and utility routines. These routines are described at the end of this manual.

Defining the menu

Three steps are required in order to tailor the menu to suit a particular application. These steps are:

1) defining a global scalar type to associate a unique label with each menu node;
2) defining the menu tree, to provide a run-time description of the menu data structure;
3) writing a control procedure for each menu group, to control menu node selection at run time.

Although the definition of the menu may appear complicated, the only menu nodes that usually require modification are those corresponding to the simulation's adjustable parameters (gains, time constants, saturation functions, and spread functions). The definition of the menu for the "prey localization model" is used as an example in the following description. The reader will find it useful to refer to Fig. 50 of the Operator's Manual for a pictorial definition of this menu.

Table 6 - Simulation System Programs and Modules.

File Name	Type	Description
LENS.PEN	**environment**	System globals
UTILITY.PEN	**environment**	utility routines interface
ENCODE.PEN	**environment**	encode procedure interface
FASTCRSR.PEN	**environment**	cursor acceleration interface
PLOT.PEN	**environment**	frame-buffer scanning interface
DIGR.PEN	**environment**	DIGR routines interface
GRINNELL.PEN	**environment**	Grinnell routines interface
LENSDEFS.PIF	**include**	Define system constants, types, and variables
LENSPROC.PIF	**include**	Define system global procedures and functions
BATCHDEFS.PIF	**include**	Define batch constants, types and variables
LENSCTINI.PIF	**include**	Menu run-time initialization and definition code
LENS.PAS	**program**	Simulation system main program
inherits environment:	UTILITY.PEN	
program name:	*lens*	
creates environment:	LENS.PEN	
includes files:	LENSDEFS.PIF, BATCHDEFS.PIF, LENSPROC.PIF	
LENSBATCH.PAS	**module**	Routines to read and execute object script
module name:	*lensbatch*	
inherits environments:	LENS.PEN, UTILITY.PEN, FASTCRSR.PEN, ENCODE.PEN	
LENSBUGS.PAS	**module**	Bug manipulation
module name:	*lensbugs*	
inherits environments:	LENS.PEN, UTILITY.PEN, FASTCRSR.PEN, GRINNELL.PEN	
LENSCTINI.PAS	**module**	Initialize and construct run-time menu
module name:	*lenscontrols_init*	
inherits environments:	LENS.PEN, UTILITY.PEN	
includes file:	LENSCTINI.PIF	

LENSCTRLS.PAS **module** Menu interaction
module name: *lenscontrols*
inherits environments: LENS.PEN, UTILITY.PEN, FASTCRSR.PEN, PLOT.PEN, GRINNELL.PEN

LENSDIGR.PAS **module** DIGR interface and display selection
module name: *lensdigrinterface*
inherits environments: LENS.PEN, UTILITY.PEN, FASTCRSR.PEN, DIGR.PEN, GRINNELL.PEN

LENSDSPTY.PAS **module** External/disparity coordinate translation
module name: *lensdisparity*
inherits environments: LENS.PEN, UTILITY.PEN, DIGR.PEN, GRINNELL.PEN

LENSFENCE.PAS **module** Fence manipulation
module name: *lensfence*
inherits environments: LENS.PEN, UTILITY.PEN, FASTCRSR.PEN, GRINNELL.PEN

LENSFIELD.PAS **module** Graphics for arena display
module name: *lensfield*
inherits environments: LENS.PEN, UTILITY.PEN, FASTCRSR.PEN, GRINNELL.PEN

LENSMODEL.PAS **module** Simulation model and support
module name: *lensmodel*
inherits environments: LENS.PEN, UTILITY.PEN, FASTCRSR.PEN, ENCODE.PEN, DIGR.PEN, FASTCRSR.PEN

LENSOVRLY.PAS **module** Graphics for OVERLAY mode
module name: *lensoverlay*
inherits environments: LENS.PEN, UTILITY.PEN, DIGR.PEN, GRINNELL.PEN

LENSPLANE.PAS **module** Scan input scene into 2-D array and display
module name: *lensplane*
inherits environments: LENS.PEN, UTILITY.PEN, DIGR.PEN

LENSRTINA.PAS **module** Scan input scene onto retinas and display
module name: *lensretina*
inherits environments: LENS.PEN, UTILITY.PEN, DIGR.PEN

LENSSAT.PAS **module** Modify and display saturation functions
module name: *lenssaturate*
inherits environments: LENS.PEN, UTILITY.PEN, FASTCRSR.PEN,

		ENCODE.PEN, GRINNELL.PEN
LENSSPRD.PAS	**module**	Modify and display spread functions
module name:		*lensspread*
inherits environments:		LENS.PEN, UTILITY.PEN, FASTCRSR.PEN, ENCODE.PEN, GRINNELL.PEN
LENSVIEW.PAS	**module**	Accommodation plane construction
module name:		*lensview*
inherits environments:		LENS.PEN, UTILITY.PEN, FASTCRSR.PEN
BATCHSYS.PAS	**program**	Script Description Language translator
program name:		*batch_system*
inherits environment:		UTILITY.PEN
includes files:		LENSDEFS.PIF, BATCHDEFS.PIF

The scalar type *command*, defined in the file LENSDEFS.PIF, associates a Pascal label with each menu node. Fig. 56 shows the definition of *command* used for the "prey localization model" simulation. The scalars for a particular menu group *must* be defined consecutively, and their order must correspond to the order in which the nodes of the group will be displayed on the graphics screen; the node that will appear at the bottom of the screen must be listed first. The Pascal labels need not match the menu node keyword used by SDL, but should be similar for clarity (compare Fig. 56 with Fig. 50). The first group defined in *command* must be the group at the top of the menu tree. Its definition must start with the label *halt*. The order of placement of the other menu groups is arbitrary. Except for the first group, the first entry for a menu group must be a unique label corresponding to that group's RESET node. The second entry for each menu group must be a unique label corresponding to that group's RECORD node. The last entry in the definition of *command* must be the label *null*.

```
type command =
    (halt, record1, retrieve, save, init1, simulate, fixate,
                            view, fences, bugs,
     reset2, record2, refresh, display, params, init2, initaccom,
                            step, stop, run,
     reset3, record3, satparams, spreadparams, gains, timecon,
                            dispint, deltat, maxtime,
     reset4, record4, sat_ft, sat_fb, sat_gu,
     reset5, record5, spr_wt, spr_wb, spr_wni,
     reset6, record6, gain_ktb, gain_kat, gain_kbu, gain_kub,
     reset7, record7, tc_ta, tc_tt, tc_tb, tc_tu,
     reset8, record8, plane, three_d, ovrlay, dsprty, world,
                            monocular, binocular, film,
     reset9, record9, pct_prism, pct_lens, pct_error, pct_sigma,
     null);
```

Fig. 56 - Example Definition of *command* Data Type.

Changes made to the type *command* must be reflected in the definitions of the three functions *is_parent*, *is_record*, and *is_reset*, found in the file LENSCTINI.PIF. These functions take a single argument of type *command* and return the Boolean value **true** just in case the node corresponding to this argument is of the type indicated by the function name; e.g. *is_parent* (*save*) should return **false** but *is_parent* (*view*) should return **true**.

The data structure defining the menu tree is constructed at run time by the procedure *initcontrols*, found in the file LENSCTINI.PIF. The global variable *controlcontext* corresponds to the root of the menu tree. It must be assigned a pointer to a record of type *controlcontextdescriptor* that contains a description of the highest-level menu group. This record, in turn, contains an array *commandset* of records of type *commanddescriptor*. The elements of this array correspond with nodes in the group; element [0] corresponds with the first node in the group, element [1] with the second node, etc. This array must be filled in with

1) the label, of type *command*, corresponding with the node,

2) the text to be displayed with the node on the menu display (this is also the SDL keyword for the menu node),

3) a pointer to another record of type *commanddescriptor* (for parent nodes) or a nil pointer (for leaf nodes).

In turn, each new *commanddescriptor* record pointed to by the parent nodes of this group must be defined. This process is repeated until the entire tree is completed. Routines *createnewcontext*, *pushparent*, and *popparent* are available from within *initcontrols* to facilitate the coding of this process. The reader is referred to the source listings for further details.

Each menu group must have a *control procedure* associated with it to control menu node selection at run-time. As an example, Fig. 57 shows the control procedure associated with the menu group descending from the menu node PARAMS. This procedure consists mainly of a repeat loop that 1) calls the procedure *commandquery* to determine the next selected menu node, 2) executes a case statement to service that node, and 3) terminates when the RESET node for the group is selected. The calls to *changerealvalue* for cases *maxtime*, *deltat*, and *dispint* update the scalar parameters *tmax*, *delt*, and *displayinterval*, to implement the menu functions MAXT, DELTAT, and INTERVAL. Since menu nodes TC, GAINS, SPREAD, and SAT are parent nodes, cases *timecon*, *gains*, *spreadparams*, and *satparams* take no action but to call the appropriate menu group control procedure associated with the selected node (i.e. *timecon* calls *selecttimeconstants*). There is no entry in the case statement for the menu node RECORD, since the procedure *commandquery* handles this selection internally. In this case, a return is not made to the calling procedure until the screen-recording function is completed, and a new menu item is selected.

```
  procedure selectparameters;

    begin
      clearworkspace;               {clear the non-menu portion of the display}

      repeat
        commandquery;               {determine next menu node selection}
        case commandtype of         {commandtype indicates selected node}

maxtime:  begin                     {menu node TMAX}
            changerealvalue (tmax, 'TMAX        ');
            fixcursor
          end;

deltat:   begin                     {menu node DELTAT}
            changerealvalue (delt, 'DELTA T     ');
            fixcursor
          end;

dispint:  begin                     {menu node INTERVAL}
            changerealvalue (displayinterval, 'DISP INTVL');
            fixcursor
          end;

timecon:  selecttimeconstants;      {menu node TC}

gains:    selectgains;              {menu node GAINS}

spreadparams: selectspread;         {menu node SPREAD}

satparams: selectsat;               {menu node SAT}

reset3:                             {menu node RESET}
        end
      until commandtype = reset3
    end;
```

Fig. 57 - Example Menu Group Control Procedure.

Defining the displays

The user must write a procedure for each unique display to be produced by the system. These display procedures are called by the procedures *updatedisplay* and *refreshdisplay* in module *lensmodel*. *updatedisplay* is activated whenever the simulation is running and the simulation time reaches the next display interval. *refreshdisplay* is activated whenever menu node REFRESH is selected. These routines will usually be identical except for certain details having to do with producing animated films (these details are obvious in the listings). The process of writing the display procedures, and modifying the routines that call them, is simplified by the use of the DIGR [Morse, 1981] high-level display generation system, and by the availability of several global switches.

DIGR uses a group of internal data structures called *DIGR data groups* to store information relating to a particular display. For each display to be produced by DIGR the user must define and "fill in" the display's data group. Once this is done, the data group's name can be used as a parameter when making calls to DIGR's display drivers. Data group names are simply integers. Therefore, it is good practice to provide a symbolic label for each data group name by defining an appropriate set of global integer constants in the file LENSDEFS.PIF.

The procedure *builddigrdatagroups*, in the file LENSDIGR.PAS, provides the run-time definition of the data groups, and must be tailored to the user's specific requirements. Only a portion of the Grinnell screen is available for drawing simulation displays. The position and size of the available area is different for batch and interactive modes. In interactive mode, two different areas are used depending on whether or not the system is in FILM mode (see Operator's Manual subsection: **Controlling the displays**). Within *builddigrdatagroups*, the integer variables *screenx0* and *screeny0* are provided to define the x and y coordinates of the lower left-hand corner of the available area, and the variables *screenwidth* and *screenheight* are provided to define its width and height. Further details on the use of DIGR are provided in the DIGR manual [Morse, 1981], and in the system listings (see especially **module** *lensdigrinterface*).

Several Boolean variables are used to define the state of the display control switches (see Operator's Manual subsection: **Controlling the displays**). Variables *three_d_display* and *overlay_display* determine the state of the tri-modal switch that governs which method is to be used to display two-dimensional arrays. If *three_d_display* is **true** then displays should be done using contour plots (DIGR routine *dd_3d_graph*), if *overlay_display* is **true** then displays should be done using overlay mode (system routine *drawfieldoverlay*, in **module** *lensoverlay*), and if neither variable is **true** then displays should be done using intensity encoding (DIGR routine *dd_inten_array*). Variables *worlddisplay* and *disparitydisplay* indicate the state of the bi-modal switch that determines which coordinate system should be used for the display. If *worlddisplay* is **true** then internal planes should be transformed into the external (x,y) coordinate system before being displayed. The routine *rebuildvisualplane*, in **module** *lensdisparity*, is provided to perform this conversion. If *disparitydisplay* is **true** then internal coordinates should be used. The values of these two variables are mutually complimentary. Variable *monocular_mode* indicates the state of the bi-modal switch that determines which internal plane is to be used if external (x,y) coordinates are being used for the display. If *monocular_mode* is true then the internal accommodative planes should be used, if *monocular_mode* is false then the disparity planes should be used. The variable *film_mode* is true if the system is in FILM mode.

Defining the simulation

System module *lensmodel* contains several procedures which must be updated in order to incorporate the simulation of a particular model into the system. This section describes the required modifications.

The actual simulation is defined by the procedure *Mymodel*. To install a new simulation, the user must update the main body of this procedure. The standard form of this procedure has an initialization section, to set initial conditions for the model's state variables, and a simulation loop, that iteratively performs the simulation calculations. Real global variable *t* specifies the current simulation clock time, *delt* specifies the time-step size, *displayinterval* specifies the time period between displays, and *tmax* specifies the maximum simulation time. Upon entry, *Mymodel* computes the number of iterations required to reach the next display time. The simulation calculation loop iterates exactly this number of times. Procedures *runmodel* and *stepmodel* call *Mymodel* repetitively, under control of the system menu.

Besides supplying the simulation procedure *Mymodel*, the user must update the simulation initialization procedure *restartmodel*. The menu group containing the RUN, STOP, and STEP nodes also contains a node labeled INIT. The procedure *restartmodel* is called whenever this node is selected. It should set the simulation time *t* to 0.0, establish initial conditions for all simulation variables, and set the

Boolean variable *initmodelsw* to true.

Four menu control procedures must be updated to accommodate the particular parameters required by the simulation. The procedure *selecttimeconstants* is activated when menu node TC is selected. It should be updated to incorporate the time constants in the new simulation. Similarly, the procedure *selectgains* is activated when GAINS is selected and should be updated to incorporate the new simulation gain parameters. Two other procedures *selectspread* and *selectsat*, residing in modules *lensspread* and *lenssaturate*, are activated, respectively, by SPREAD and SAT and should be updated to handle the simulation's spread and saturation functions.

Preprogrammed initial values for all simulation parameters are established by the procedure *initmodel*. This procedure is called at the start of the simulation session, and is also called when the INIT node at the top level of the menu is selected.[4] The procedures *savemodel*, *savespread* (in module *lensspread*), and *savesat* (in module *lenssaturate*) are called when menu node SAVE is selected. These procedures write the various simulation parameters to a disk file for later restoration. The corresponding procedures *restoremodel*, *restorespread*, and *restoresat* restore parameters from the disk file when menu node FETCH is selected. All of these procedures must be updated to accommodate the parameters of the new simulation.

[4] Note that this INIT node is different from the INIT node in the menu group containing RUN, STOP, and STEP.

Compiling and linking the system

The system is broken down into several compilable programs and modules so that small changes made to one section of code do not necessitate a recompilation of the entire system, but simply a recompilation of the modified module. Programs and modules to be compiled by the VAX Pascal compiler have file names ending in ".PAS." Object files from the Pascal compilations are placed in the object library LENSLIB.OLB for later access by the linker. The command stream depicted in Fig. 58, and contained in the file LPAS.COM, compiles an individual module, updates the object library, and does some clean-up. The single parameter required by this command stream is the file name of the file to be compiled.

```
$!                          LPAS.COM
$!   Compile Source Module and Install in System Object Library
$!
$ set default [neudhh.frog.lens]        ! establish default directory
$
$ pascal/list/noopt/nodebug 'p1         ! compile module with listing
$
$ libr lenslib 'p1                      ! install object in library
$ delete 'p1'.obj;*                     ! delete the object
$ purge 'p1'.*                          ! and clean up old sources and listings
$!
$ exit                                  ! End of Job
```

Fig. 58 - Command Stream to Compile an Individual Module.

The interactive and batch versions of the system are created from the same object library. They differ only in the set of graphics routines to which they are linked. The interactive system is linked with the standard Grinnell library routines [Morse, 1979], and the batch system is linked with the GUS Grinnell emulator routines [Cromarty & Sutton, 1983]. The two command streams shown in Fig. 59 are contained in the files LINKLENS.COM and GUSLENS.COM. LINKLENS links the interactive system, and GUSLENS links the batch system. The resulting executable images are LENS.EXE and BATCHLENS.EXE.

```
$!                    LINKLENS.COM
$! Link Interactive System: use standard Grinnell graphics library
$!
$ set default [neudhh.frog.lens]        ! establish default directory
$!
$ link lenslib/lib/nomap/nodebug/executable=lens/include=lens,-
[neudhh.pascal]utillib/lib,[neudhh.graphics]graphlib/lib,-
[neudhh.graphics]small/lib,[neudhh.graphics]glib/lib,-
[morse]digr/lib
$!
$ exit                                  ! End of Job


$!                     GUSLENS.COM
$! Link Batch System: use GUS Grinnell emulator graphics library
$!  valid GUS devices are Metafile, Grinnell, terminal
$!
$ set default [neudhh.frog.lens]        ! establish default directory
$
$ grguslink [neudhh.frog.lens]lenslib/nomap/nodebug/lib/include=lens-
/executable=batchlens,-
[neudhh.pascal]utillib/lib,[neudhh.graphics]graphlib/lib,-
[neudhh.graphics]small/lib,[neudhh.graphics]glib/lib,-
[morse]digr/lib metafile gr vt52terms
$!
$ exit                                  ! End of Job
```

Fig. 59 - Command Streams to Link Interactive and Batch Versions of System.

Whenever the main program file LENS.PAS (or any of its *include* files: LENSDEFS.PIF, LENSPROC.PIF, BATCHDEFS.PIF) is modified, it is necessary to recompile and relink the entire system. This is because when LENS.PAS is compiled a new environment file LENS.PEN is also created. Attempting to link the system without compiling its modules against this new environment will result in link-time warnings and may result in an inoperable system. The command stream shown in Fig. 60 may be used to facilitate recompilation and linking of the entire system. This command stream resides in the file NEWLENS.COM. At the end of this command stream are commands to compile and link the SDL translator program BATCHSYS. BATCHSYS uses the *include* files LENSDEFS.PIF, BATCHDEFS.PIF, and LENSCTINI.PIF and thus must be recompiled and linked whenever changes are made to any of these files.

Simulation system procedures and functions

This section provides a brief description of a selected group of simulation system procedures and functions. Routines selected for description are those that are likely to either require modification or be directly used by the programmer who is configuring the system.

```
$!                  NEWLENS.COM
$!   Compile and Link all System Modules and Programs
$!
$ set default [neudhh.frog.lens]        ! establish default directory
$!
$ lpas := @lpas.com                     ! define special compile command
$!
$! compile all modules and place in system library
$!
$ lpas lens                             ! compile first to build environment file
$ lpas lensbatch
$ lpas lensbugs
$ lpas lensctrls
$ lpas lensctini
$ lpas lensdigr
$ lpas lensdspty
$ lpas lensfence
$ lpas lensfield
$ lpas lensmodel
$ lpas lensovrly
$ lpas lensplane
$ lpas lensrtina
$ lpas lenssat
$ lpas lenssprd
$ lpas lensview
$!
$!   Delete old executables and link to make new ones
$!
$ delete lens.exe;*                     ! delete old interactive version
$ @linklens                             ! interactive: link Grinnell library routines
$ delete batchlens.exe;*                ! delete old batch version
$ @guslens                              ! batch: link with GUS Grinnell emulator
$!
$!   Compile and link the SDL translator program BATCHSYS
$!
$ delete batchsys.exe;*                 ! get rid of old executable
$ pas/lis batchsys                      ! compile translator
$ link batchsys,-
[neudhh.pascal]utillib/lib,[neudhh.graphics]graphlib/lib,-
[neudhh.graphics]small/lib,[neudhh.graphics]glib/lib
$ delete batchsys.obj;*                 ! get rid of object
$!
$ exit                                  ! End of Job
```

Fig. 60 - Command Stream to Compile and Link Entire System.

batcherror.

Purpose: write an error message to the batch Run Log file.

Pascal definition: **procedure** batcherror (errormsg: message);

Source file: LENSBATCH.PAS

Description: The text contained in *errormsg* is written to the batch Run Log file, and the count of batch run-time errors in the global variable *errorcounter* is incremented. The data type *message* is defined **packed array** [1..30] **of char**.

nextentry.

Purpose: obtain the next entry in the batch script.

Pascal definition: **function** nextentry: entrytype;

Source file: LENSBATCH.PAS

Description: When running in batch mode the system reads the entries in the object version of the script into consecutive elements of the array *script*. A global variable *spc* is the array index of the currently active script entry. *nextentry* increments *spc* and retrieves the next script entry. If this entry corresponds with either a menu node keyword, a number, a string, or the SDL $\wedge$ statement *nextentry* returns a corresponding scalar (either *menuitem*, *textstring*, *number*, or *up*). If the next entry corresponds with the SDL "LOOP" statement the loop counter is initialized and *spc* is adjusted to point to the first entry following the "LOOP" entry. If the next entry corresponds with the SDL "ENDLOOP" statement the loop counter is updated and a test for loop completion is performed. If further iteration is required *spc* is adjusted to point to the first entry following the corresponding "LOOP" entry. If no further iteration is required *spc* is adjusted to point to the first entry following the "ENDLOOP" entry. The procedure, indicated above, iterates until an entry of type *menuitem*, *textstring*, *number*, or *up* is encountered and a return is made. If the end of the script is reached before a return is possible an error message is written to the Run Log, and *up* is returned.

updaterealvalue.

Purpose: change the value of a real variable using a numeric argument from the batch script.

Pascal definition: **procedure** updaterealvalue (var x: real);

Source file: LENSBATCH.PAS

Description: The value of *x* is updated based on the current entry in the batch script. The calling procedure must check to make sure that this entry, *script* [*spc*], is of type *number* before calling *updaterealvalue*. The number from the script will be used to either replace, increment, or decrement *x* based upon the type of the entry in the original source script.

changerealvalue.

Purpose: change the value of a real variable using either the interactive graphics display and joystick or a numeric argument from the batch script.

Pascal definition: **procedure** changerealvalue (var x: real; description: alpha);

Source file: LENSBATCH.PAS

Description: If the system is running in interactive mode the routine *pickrealvalue* (see subsection: **Utility procedures and functions**) is called to display and change the value of *x* using the graphics display and joystick. If the system is running in batch mode the function *nextentry* (see above) is invoked to determine the next entry in the batch script. If the next entry is of type *number* then *updaterealvalue* (see above) is called to change the value of *x*. If the next entry is not of type *number* an error message is written to the Run Log, and return is made with *x* unchanged.

is_parent.

Purpose: test for parent menu node.

Pascal definition: **function** is_parent (cmnd: command): Boolean;

Source file: LENSCTINI.PIF

Description: This function returns **true** if *cmnd* corresponds with an internal (parent) menu node, and returns **false** if it does not. This routine must be updated whenever changes are made to the structure of the menu.

is_record.

Purpose: test for record menu node.

Pascal definition: **function** is_record (cmnd: command): Boolean;

Source file: LENSCTINI.PIF

Description: This function returns **true** if *cmnd* corresponds with a RECORD menu node, and returns **false** if it does not. This routine must be updated whenever changes are made to the structure of the menu.

is_reset.

Purpose: test for reset menu node.

Pascal definition: **function** is_reset (cmnd: command): Boolean;

Source file: LENSCTINI.PIF

Description: This function returns **true** if *cmnd* corresponds with a RESET menu node, and returns **false** if it does not. This routine must be updated whenever changes are made to the structure of the menu.

initcontrols.

Purpose: run-time construction of menu tree.

Pascal definition: **procedure** initcontrols;

Source file: LENSCTINI.PIF

Description: This procedure builds the data structure representing the menu tree. It must be updated whenever changes are made to the structure of the menu.

fixcursor.

Purpose: position cursor in the center of the menu display and deselect current menu node.

Pascal definition: **procedure** fixcursor;

Source file: LENSCTRLS.PAS

Description: If the system is in interactive mode the current menu node selection is terminated by changing the color of the disk and text associated with that node to black, and changing the value of *commandtype* to *null*. The cursor is moved to the center of the menu display if it is not currently positioned on the menu display. In batch mode, this procedure does nothing. *fixcursor* is called by the various menu control procedures when a menu function is completed.

clearworkspace.

Purpose: clear the portion of the display screen not devoted to the menu display.

Pascal definition: **procedure** clearworkspace;

Source file: LENSCTRLS.PAS

Description: In interactive mode, this procedure clears the non-menu portion of the display screen. In batch mode, it clears the entire virtual display, erasing any graphics drawn since the last call to write the virtual image to the "Metafile" (*mf_mark_file*, see GUS manual [Cromarty & Sutton, 1983]).

commandquery.

Purpose: get the next menu node selection.

Pascal definition: **procedure** commandquery;

Source file: LENSCTRLS.PAS

Description: The global variable *commandtype* is updated to indicate a new menu node selection. When this procedure is activated in interactive mode *commandtype* is updated by monitoring the cursor position, and the ENTER and HOME buttons on the Grinnell's control box. If an internal (parent) node is selected the menu group descending from the selected node is displayed before returning to the calling procedure. When this procedure is activated in batch mode *commandtype* is updated by using the function *nextentry* (see above) to retrieve the next entry in the script. In either interactive or batch modes, selection of a RECORD menu node will cause *commandquery* to perform the RECORD function to save the graphics display, and then look for the next selection. A call to *commandquery* is normally the first statement of the **repeat** loop forming the main body of a menu control procedure.

builddigrdatagroups.

Purpose: run-time definition of DIGR data groups.

Pascal definition: **procedure** builddigrdatagroups;

Source file: LENSDIGR.PAS

Description: This routine is mainly user written. It must make all of the calls to the DIGR data-group definition routines necessary to define each of the data groups to be used in producing simulation displays.

displaydisparityplane.

Purpose: display internal field(s) in the internal coordinate system.

Pascal definition: **procedure** displaydisparityplane;

Source file: LENSDIGR.PAS

Description: This routine is user written, and provides the appropriate call(s) to DIGR display routines to display an internal field in the internal coordinate system.

builddisparityplane.

Purpose: construct an internal field from the images on the two retinas.

Pascal definition: **procedure** builddisparityplane;

Source file: LENSDSPTY.PAS

Description: This routine is null in the simulation of the "prey localization model" but is the appropriate place for the user to insert code to construct a field in the internal (retinal angle vs. disparity) coordinate system from two retinal images. In the "cue interaction model" simulation, for example, this routine performs the retinal disparity calculations described by the equations in Appendix B.

rebuildvisualplane.

Purpose: map an internal field back onto the external coordinate system.

Pascal definition: **procedure** rebuildvisualplane (var D: disparityplane; var R: retinalimage; var P: featureplane);

Source file: LENSDSPTY.PAS

Description: The array *D* must contain the matrix representation of a field in the internal (retinal angle vs. disparity) coordinate system. The array *R* must contain the vector representation of the retinal image on the same side of the visual system as the field represented by *D*. The array *P* is a matrix that represents a field in the external (x,y) coordinate system. *rebuildvisualplane* does the appropriate coordinate conversion to map each element of *D* onto the corresponding element of *P*, using the coordinate conversion equations shown in Appendix B.

Mymodel.

Purpose: perform iterative simulation calculations.

Pascal definition: **procedure** Mymodel;

Source file: LENSMODEL.PAS

Description: The main body of this routine is user supplied. Details are provided in subsection: **Defining the simulation.**

updatedisplay.

Purpose: issue simulation display requests after each display interval while running the model.

Pascal definition: **procedure** updatedisplay;

Source file: LENSMODEL.PAS

Description: The main body of this routine is user supplied. Details are provided in subsection: **Defining the simulation.**

refreshdisplay.

Purpose: issue simulation display requests when REFRESH is selected on the system menu.

Pascal definition: **procedure** refreshdisplay;

Source file: LENSMODEL.PAS

Description: The main body of this routine is user supplied. Details are provided in subsection: **Defining the simulation.**

restartmodel.

Purpose: initialize the simulation when INIT is selected on the system menu.

Pascal definition: **procedure** restartmodel;

Source file: LENSMODEL.PAS

Description: The main body of this routine is user supplied. Details are provided in subsection: **Defining the simulation.**

selectgains.

Purpose: menu control procedure for updating simulation gains.

Pascal definition: **procedure** selectgains;

Source file: LENSMODEL.PAS

Description: The main body of this routine is user supplied. Details are provided in subsection: **Defining the simulation.**

selecttimeconstants.

Purpose: menu control procedure for updating simulation time constants.

Pascal definition: **procedure** selecttimeconstants;

Source file: LENSMODEL.PAS

Description: The main body of this routine is user supplied. Details are provided in subsection: **Defining the simulation.**

savemodel.

Purpose: save simulation gains, time constants, and control parameters in the save file.

Pascal definition: **procedure** savemodel;

Source file: LENSMODEL.PAS

Description: The main body of this routine is user supplied. Details are provided in subsection: **Defining the simulation.**

restoremodel.

Purpose: restore simulation gains, time constants, and control parameters from the save file.

Pascal definition: **procedure** restoremodel;

Source file: LENSMODEL.PAS

Description: The main body of this routine is user supplied. Details are provided in subsection: **Defining the simulation.**

initmodel.

Purpose: initialize the simulation's gains, time constants, saturation functions, spread functions, and control parameters at the start of the simulation run and when INIT is selected at the top level of the system menu.

Pascal definition: **procedure** initmodel;

Source file: LENSMODEL.PAS

Description: The main body of this routine is user supplied. Details are provided in subsection: **Defining the simulation.**

drawfieldoverlay.

Purpose: draw the display when the display modes switches are set to WORLD and OVERLAY.

Pascal definition: **procedure** drawfieldoverlay (datagroup: integer);

Source file: LENSOVRLY.PAS

Description: This procedure may be called by any of the various user supplied display routines. Normally, a display routine will check the global Boolean variables *worlddisplay* and *overlay_display* to assure that WORLD and OVERLAY have been selected before calling *drawfieldoverlay*. The user must update the procedure *ovlshowmodelstate*, that is nested within *drawfieldoverlay*, so that it will draw a picture of the current model's state (in external coordinates) to the area of the display screen defined in DIGR data group *datagroup*. Before calling *ovlshowmodelstate*, *drawfieldoverlay* draws a correctly scaled and positioned version of the input scene (the bugs and fences) in the same area of the display screen. The resulting display will show the model's state superimposed upon an image of the input scene.

selectsat.

Purpose: menu control procedure for updating simulation saturation functions.

Pascal definition: **procedure** selectsat;

Source file: LENSSAT.PAS

Description: The main body of this routine is user supplied. Details are provided in subsection: **Defining the simulation.**

savesat.

Purpose: save simulation saturation-function descriptions in the save file.

Pascal definition: **procedure** savesat;

Source file: LENSSAT.PAS

Description: The main body of this routine is user supplied. Details are provided in subsection: **Defining the simulation.**

restoresat.

Purpose: restore simulation saturation-function descriptions from the save file.

Pascal definition: **procedure** restoresat;

Source file: LENSSAT.PAS

Description: The main body of this routine is user supplied. Details are provided in subsection: **Defining the simulation.**

selectspread.

Purpose: menu control procedure for updating simulation spread functions.

Pascal definition: **procedure** selectspread;

Source file: LENSSPRD.PAS

Description: The main body of this routine is user supplied. Details are provided in subsection: **Defining the simulation.**

savespread.

Purpose: save simulation spread-function descriptions in the save file.

Pascal definition: **procedure** savespread;

Source file: LENSSPRD.PAS

Description: The main body of this routine is user supplied. Details are provided in subsection: **Defining the simulation.**

restoresprd.

Purpose: restore simulation spread-function descriptions from the save file.

Pascal definition: **procedure** restoresprd;

Source file: LENSSPRD.PAS

Description: The main body of this routine is user supplied. Details are provided in subsection: **Defining the simulation.**

Interface with Grinnell, DIGR, and GUS routines

Graphics are produced by making calls to routines in the Grinnell [Morse, 1979] and DIGR [Morse, 1981] libraries. These graphics-support routines are written in FORTRAN. Interfacing them with the Pascal routines of the simulation system is simplified by the use of a set of "plug" routines. For each subroutine in the Grinnell and DIGR libraries a corresponding Pascal procedure was written to perform any required argument conversions and then call the FORTRAN subroutine. The name of the Pascal "plug" routine is identical to the name of the FORTRAN routine but with any underscore (_) characters removed. For example, *grvectclip* is the "plug" routine for the Grinnell routine *gr_vect_clip*. The order and type of the arguments used in calling the "plug" routines are identical with those required by the FORTRAN routines. The only restriction is that actual character-string parameters must be of type:

packed array [1..10] **of char**.

Source code for the "plug" routines is contained in the files GRINNELL.PAS and DIGR.PAS, object files are kept in UTILLIB.OLB, and the corresponding Pascal environment files are GRINNELL.PEN and DIGR.PEN.

The Grinnell emulator mode of the GUS interactive graphics system [Cromarty & Sutton, 1983] is used by the batch version of the system to produce all graphics. The Grinnell emulator makes the use of GUS invisible to the user except during linking of the system. The use of GUS provides a degree of graphics device independence and allows the "metafile" pseudo-device to be used to save graphic images.

Utility procedures and functions

During the development of the simulation system, various general procedures and functions were written and collected together in the object library UTILLIB.OLB. Although these routines have applicability outside of the simulation system, they were used liberally in the simulation software and are not documented elsewhere. Therefore, a brief description of each of these routines is provided below. For each source file *SOURCE*.PAS listed in these descriptions there is a corresponding Pascal environment file *SOURCE*.PEN. Several of these routines take arguments of type *vector* or *alpha*. The argument type *vector* is a general purpose type used to define formal parameters which correspond to actual parameters that are one- or two-dimensional real arrays in the calling program. *vector* is typed

array [0..262143] **of** real.

A particular routine's assumptions about the type of the actual parameter corresponding to a formal parameter of type *vector* are noted in the routine's description. Arguments of type *alpha* are used to specify character-string parameters. *alpha* is typed

packed array [1..10] **of** char.

encode.

Purpose: convert a real number to an ASCII string.

Pascal definition: **procedure** encode (x: real; var string: alpha; n: integer);

Source file: ENCODE.PAS

Description: *x* is converted to an ASCII string of length *n*, and stored in *string*.

grfastcrsr.

Purpose: wait for joystick movement and accelerate the cursor.

Pascal definition: **procedure** grfastcrsr (var x, y: integer; var homeflag: Boolean);

Source file: FASTCRSR.PAS

Description: This routine does not return until either the cursor is moved or HOME followed by ENTER are depressed on the Grinnell control box. On return, *x* and *y* hold the new cursor coordinates (on a scale of 0–511). *homeflag* will be set **true** if HOME-ENTER is depressed. This routine requires support subroutines contained in the libraries GRAPHICS.OLB, GLIB.OLB, and SMALL.OLB.

graccelcrsr.

Purpose: accelerate cursor movement.

Pascal definition: **procedure** graccelcrsr (var x, y: integer; var enterflag, homeflag: Boolean);

Source file: FASTCRSR.PAS

Description: This routine returns only after the ENTER button on the Grinnell control box is depressed. While this routine is active cursor movement in response to the joystick is accelerated. Upon return, *x* and *y* hold the new cursor coordinates (on a scale of 0–511). *enterflag* will be set to **true**. *homeflag* will be set to **true** if the HOME button was depressed immediately prior to the ENTER button. This routine requires support subroutines contained in the libraries GRAPHICS.OLB, GLIB.OLB, and SMALL.OLB.

pickrealvalue.

Purpose: modify a real-valued scalar using the joystick.

Pascal definition: **procedure** pickrealvalue (var realnum: real; varlabel: alpha);

Source file: FASTCRSR.PAS

Description: The value of *realnum* is indicated on the graphics screen by the height and color of a rectangular bar. A blue bar is used to indicate a positive value and a red bar to indicate a negative value. The text in *varlabel* is used as a legend, and the numeric value of the scalar is displayed below the rectangle. The scalar's value may be increased by using the joystick to raise the height of the bar, and decreased by lowering the height of the bar. The routine exits when HOME followed by ENTER are depressed on the Grinnell's control box. This routine requires support subroutines contained in the libraries GRAPHICS.OLB, GLIB.OLB, and SMALL.OLB.

pickrealarray.

Purpose: modify an array of real-valued scalars using the joystick.

Pascal definition: **procedure** pickrealarray (var realarray: vector; n: integer; minval, maxval: real; varlabel: alpha; xmin, ymin, xmax, ymax: integer);

Source file: FASTCRSR.PAS

Description: This routine operates like *pickrealvalue* except that it can be used to modify each of the elements of an array of reals. Each time the ENTER button on the Grinnell's control box is depressed the next consecutive element of the array is displayed for modification. The routine exits when HOME followed by ENTER are depressed. *xmin*, *ymin*, *xmax*, and *ymax* specify the lower left and upper right Grinnell screen coordinates of the window which will be used for the interactive display. *minval* and *maxval* specify minimum and maximum values to be used for initial scaling of the display. If these are set to zero (0) scaling will be done relative to the initial value of the variable being modified. This routine requires support subroutines contained in the libraries GRAPHICS.OLB, GLIB.OLB, and SMALL.OLB.

fatarrow.

Purpose: draw a large block arrow on graphics screen, with specified center point.

Pascal definition: **procedure** fatarrow (x, y, dx, dy, r, g, b: integer);

Source file: FASTCRSR.PAS

Description: *x* and *y* specify the Grinnell screen coordinates of the center of the arrow. *dx* and *dy* specify the horizontal and vertical components of the vector difference between the starting and ending points of the arrow. Thus, they determine its length, slope, and orientation. Its width is proportional to its length. This routine requires support subroutines contained in the libraries GRAPHICS.OLB, GLIB.OLB, and SMALL.OLB.

drawarrow.

Purpose: draw an arrow on the graphics screen, with specified center point.

Pascal definition: **procedure** drawarrow (x, y: integer; slope, Lsqr: real; r, g, b: integer);

Source file: FASTCRSR.PAS

Description: *x* and *y* specify the Grinnell screen coordinates of the center of the arrow. *slope* specifies the slope of the arrow, and *Lsqr* specifies the signed square of the arrow's length, i.e. the square of the length in pixels, with sign to indicate the direction of the arrow with respect to its slope. This routine requires support subroutines contained in the libraries GRAPHICS.OLB, GLIB.OLB, and SMALL.OLB.

drawarrowfrompt.

Purpose: draw an arrow on graphics screen, with specified tail-point.

Pascal definition: **procedure** drawarrowfrompt (x, y: integer; slope, Lsqr: real; r, g, b: integer);

Source file: FASTCRSR.PAS

Description: *drawarrowfrompt* is identical to *drawarrow* except that *x* and *y* specify the starting-point of the arrow rather than its center. This routine requires support subroutines contained in the libraries GRAPHICS.OLB, GLIB.OLB, and SMALL.OLB.

grprintbw.

Purpose: copy graphics screen to print file.

Pascal definition: **procedure** grprintbw (threshold: integer);

Source file: PLOT.PAS

Description: This routine scans the Grinnell frame-buffer to retrieve the current graphic image. This image is converted into the correct format for printing in graphics mode on the Printronix printer, and is saved in the file PLOT.PLT. The sum of the Grinnell color values for a pixel is compared with *threshold*. Pixels with color intensity below *threshold* are not printed, and pixels above *threshold* are printed. The result is a black and white negative of the image on the Grinnell screen. During the operation of this procedure the cursor is displayed on the center of the line currently being scanned, so that the user can observe its progress. Scanning of the frame buffer takes about one minute. This routine requires a support subroutine contained in the library GLIB.OLB.

polynomial.

Purpose: evaluate a polynomial.

Pascal definition: **procedure** polynomial (var W: vector; order: integer; var coeff: vector; tlo, Dt: real; n: integer);

Source file: MATHPACK.PAS

Description: A polynomial of order *order* and determined by the coefficients in *coeff* (*coeff* [0] = a_0, *coeff* [1] = a_1, etc.) is evaluated for *n* values of its independent variable. *tlo* specifies the starting value of the independent variable, and *Dt* specifies its increment. The calculated values are stored in consecutive cells of *W*.

convolve.

Purpose: convolution either along a vector or across columns of a matrix keeping row constant.

Pascal definition: **procedure** convolve (var U, V: vector; ixmax: integer; W: vector; itmax, itlim: integer; Dx: real);

Source file: MATHPACK.PAS

Description: *W* contains sampled values of the right half of a continuous symmetric one-dimensional spread function, with values spaced apart by *Dx* units of the independent variable (*W*[0] = w(0), *W*[1] = w(Dx), *W*[2] = w(2Dx), etc.). It is assumed that *W* is typed **array** [0..*itmax*] **of real** in the calling program. *W* is taken to be zero beyond the index *itlim*. *U* contains the input vector to be convolved with *W*. *V* is the output. Upon return *V* = *W***U*. The sampling interval for both *U* and *V* is *Dx*. It is assumed that *U* and *V* are typed **array** [−*ixmax*..*ixmax*] **of real** in the calling program. Alternatively, *U* and *V* may be two-dimensional arrays of type **array** [−*ixmax*..*ixmax*, −*iymax*..*iymax*] **of real**, where the value of *iymax* is arbitrary. In this case, the convolution will be done along a single row of the array. Actual parameters, corresponsing with *U* and *V* should specify the base address of the given row; e.g. to specify the 6th row of array A use A[6,−*iymax*].

convolve_row.

Purpose: convolution across rows of matrix, keeping column constant.

Pascal definition: **procedure** convolve_row (var U, V: vector; ixmax, iymax: integer; var W: vector; itmax, itlim: integer; Dx: real);

Source file: MATHPACK.PAS

Description: This routine is similar to *convolve*, except that *U* and *V* are assumed to be two-dimensional arrays, and the convolution is performed along one column of array *U*. *U* and *V* are assumed to be typed **array** [−*ixmax*..*ixmax*, −*iymax*..*iymax*] **of real** in the calling program. Actual parameters corresponding with *U* and *V* should specify the base address of the given column; e.g. to specify the 6th column of array A use A[−ixmax, 6].

convolve_2D.

Purpose: two dimensional convolution.

Pascal definition: **procedure** convolve_2D (var U, V: vector; ixmax, iymax: integer; var W: vector; ismax, itmax, islim, itlim: integer; Dx, Dy: real);

Source file: MATHPACK.PAS

Description: This routine is similar to *convolve* but performs a two-dimensional convolution over the entire two-dimensional array *U*, with the result stored in the two-dimensional array *V*. *U* and *V* are assumed to be typed **array** [*−ixmax..ixmax*, *−iymax..iymax*] **of real** in the calling program. *W* is the matrix representation of the first quadrant of a symmetric continuous two-dimensional spread function w, and is assumed to be typed **array** [0..*ismax*, 0..*itmax*] **of real** in the calling program. *W* is taken to be zero outside of the rectangle defined by the corner points (*islim*, *itlim*) and (*−islim*, *−itlim*) *Dx* specifies the change in horizontal independent variable per increment of the row index, and *Dy* specifies the change in the vertical independent variable per increment of the column index.

integral.

Purpose: Trapezoidal Rule integration.

Pascal definition: **function** integral (var U: vector; n: integer; Dx: real): real;

Source file: MATHPACK.PAS

Description: The Trapezoidal Rule is used to compute the integral of the function represented by the vector *U*. Dx specifies the change in independent variable per unit change in the index of *U*, and n is number of points in *U*. It is assumed that *U* is typed **array** [*n*] **of real** in the calling program.

project.

Purpose: project contents of one array onto another array.

Pascal definition: **procedure** project (var Afrom, Ato: vector; ifmax, jfmax, itmax, jtmax: integer);

Source file: MATPACK.PAS

Description: This routine projects the contents of the source array *Afrom* onto the target array *Ato*. The two arrays may be of different mesh size. The contents of the source mesh points which overlay a target mesh point are averaged (weighted according to the amount of overlap) into the target point. *Afrom* is assumed to be typed **array** [*−ifmax..ifmax*, *−jfmax..jfmax*] **of real**, and *Ato* is assumed to be typed **array** [*−itmax..itmax*, *−jtmax..jtmax*] **of real** in the calling program.

rowtocolmajor.

Purpose: convert row major array to column major array.

Pascal definition: **procedure** rowtocolmajor (var R: vector; nrow, ncol: integer);

Source file: MATPACK.PAS

Description: This routine does an in-place conversion of the two-dimensional array represented by *R* from row major representation (Pascal standard) to column major representation (FORTRAN standard). This routine is useful when interfacing with subroutines written in FORTRAN. It is assumed that *R* is typed **array** [*nrow*, *ncol*] **of real** in the calling program.

coltorowmajor.

Purpose: convert column major array to row major array.

Pascal definition: **procedure** coltorowmajor (var C: vector; nrow, ncol: integer);

Source file: MATPACK.PAS

Description: This routine does an in-place conversion of the two-dimensional array represented by *C* from column major representation (FORTRAN standard) to row major representation (Pascal standard). This routine is useful when interfacing with subroutines written in FORTRAN. It is assumed that *C* is typed **array** [*nrow*, *ncol*] **of real** in the calling program.

matmpy.

Purpose: multiply matrices.

Pascal definition: **procedure** matmpy (var A: vector; n, m: integer; var B: vector; l: integer; var C: vector);

Source file: MATPACK.PAS

Description: This routine multiplies matrices *A* and *B* and stores the result in *C*. It is assumed that *A* represents an $n \times m$ real matrix, *B* an $m \times l$ real matrix, and *C* an $n \times l$ real matrix in the calling program.

max.

Purpose: determine integer maximum.

Pascal definition: **function** max (i, j: integer): integer;

Source file: UTILITY.PAS

Description: This function returns the integer maximum of *i* and *j*.

min.

Purpose: determine integer minimum.

Pascal definition: **function** min (i, j: integer): integer;

Source file: UTILITY.PAS

Description: This function returns the integer minimum of *i* and *j*.

amax.

Purpose: determine real maximum.

Pascal definition: **function** amax (a, b: real): real;

Source file: UTILITY.PAS

Description: This function returns the real-valued maximum of *a* and *b*.

amin.

Purpose: determine real minimum.

Pascal definition: **function** amin (a, b: real): real;

Source file: UTILITY.PAS

Description: This function returns the real-valued minimum of a and b.

ipow.

Purpose: raise a real number to an integer power.

Pascal definition: **function** ipow (a: real; i: integer): real;

Source file: UTILITY.PAS

Description: This function computes a^i using iterative multiplication.

pow.

Purpose: raise a real number to a real power.

Pascal definition: **function** pow (a, b: real): real;

Source file: UTILITY.PAS

Description: This function computes a^b using logarithms.

cube.

Purpose: cube a real number.

Pascal definition: **function** cube (a: real): real;

Source file: UTILITY.PAS

Description: This function computes a^3.

sign.

Purpose: determine arithmetic sign.

Pascal definition: **function** sign (a: real): integer;

Source file: UTILITY.PAS

Description: This function returns +1 if *a* is either positive or zero. It returns −1 if *a* is negative.

divide.

Purpose: protected divide.

Pascal definition: **function** divide (num, denom: real): real;

Source file: UTILITY.PAS

Description: This function does real division with divide by zero protection. It returns *num/denom* just in case **abs** (*num/denom*) $\leq 10^{15}$. It returns $10^{15}\times$*sign* (*num*)$\times$*sign* (*denom*) if **abs** (*num/denom*) $> 10^{15}$.

tan.

Purpose: triginometric tangent.

Pascal definition: **function** tan (theta: real): real;

Source file: UTILITY.PAS

Description: This function computes the triginometric tangent of the angle *theta*. *theta* must be expressed in radians.

arctangent.

Purpose: two argument triginometric arctangent.

Pascal definition: **function** arctangent (opposite, adjacent: real): real;

Source file: UTILITY.PAS

Description: This function returns the arctangent, in radians over the range $-\pi$ to π, of the angle determined by *opposite* and *adjacent*.

ceil.

Purpose: ceiling.

Pascal definition: **function** ceil (a: real): integer;

Source file: UTILITY.PAS

Description: This function returns the least integer $\geq a$.

floor.

Purpose: floor.

Pascal definition: **function** floor (a: real): integer;

Source file: UTILITY.PAS

Description: This function returns the greatest integer $\leq a$.

ran.

Purpose: generate a uniform random number.

Pascal definition: **function** ran (var seed: integer): real;

Source file: UTILITY.PAS

Description: This function returns a sample of a random variable that is uniformly distributed over the range [0,1). As a side-effect, the value of *seed* is modified.

wait_for.

Purpose: delay process for specified number of seconds.

Pascal definition: **procedure** wait_for (sec: real);

Source file: UTILITY.PAS

Description: This procedure causes control to pass to the executive, with reinitiation of the calling process after *sec* seconds. This routine requires support subroutines contained in the library GLIB.OLB.

testbatch.

Purpose: test for batch mode.

Pascal definition: **function** testbatch: Boolean;

Source file: UTILITY.PAS

Description: This function returns **true** if the calling process is running under batch. It utilizes the routine *utl_tst_batmode* which is coded in VAX MACRO and included in UTILLIB.OLB.