Reinforcement Learning of Visually Guided Spatial Goal Directed Movement

Paul Toombs

Submitted in partial fulfillment of the requirements for
the degree of

Doctor of Philosophy

27 October 1997

Supervised by Professor W. A. Phillips

Psychology Department
University of Stirling
Scotland
FK9 4LA

Acknowledgements

Bill Phillips, having taught and inspired me on his MSc course, has helped, supported and advised me at every level throughout this research. His ideas, and the breadth of his knowledge, still amaze me, and our long discussions have been one of the most enjoyable aspects of my time at Stirling. I am also grateful for his kindness, tolerance, and patience with me over the years.

Dario Floreano is responsible for getting me into this area of research, through his infectious enthusiasm and fascinating work. Since then he has constantly helped, encouraged, and passionately argued with me, and I am immensely grateful for all.

Paul Miller has helped me develop the ideas here, and consistently made me rethink most of them. I thank him for all his help, discussion, and his sense of humour.

This stuff would not have been possible without the help, advice, and discussion of folk in the Centre for Computational and Cognitive Neuroscience, from whom I have learned a great deal, and with whom I have had a great time.

Thanks to Tony Zawadzki for his enthusiasm and ability to see things from a different point of view; Steve Dakin, Trish Carlin, and Roger Watt for discussion of vision; John McGeever, for discussion of computation; and Ben Craven, Peter Hancock and Kevin Swingler for miscellaneous advice and discussion.

Thanks to Dario Floreano, Peter Hancock and John McGeever, for bravely reading, commenting upon, and improving earlier drafts of this work.

Special thanks to Will Goodall, for always being such a good friend and for making me laugh so often, and to Trish Carlin, Harry Hill, and Lawence Gerstley for making these last few years so enjoyable.

Jane Gott and Vanessa McCulluch have enriched my life beyond measure, and I am immensly grateful for their unflagging support and encouragement. Thanks to Leslie Glenn; and to Angelo Pyrsos and Yiannis Kapaios, for keeping me nearly sane, and making me laugh during the grim months of writing it all up.

I should like to dedicate this thesis, with love and gratitude, to my parents: Pat and Norman Toombs.

Abstract

A range of visually guided, spatial goal directed tasks are investigated, using a computational neuroethology approach. Animats are embedded within a bounded, 2-D environment, and map a 1-D visual array, through a convolution network, to a topography preserving motor array that stochastically determines the direction of movement. Temporal difference reinforcement learning modifies the convolution network in response to a reinforcement signal received only at the goal location.

Three forms of visual coding are compared: multiscale coding, where the visual array is convolved by Laplacian of Gaussian filters at a range of spatial scales before convolution to determine the motor array; rectified multiscale coding, where the multiscale array is split into positive and negative components; and intensity coding, where the unfiltered visual array is convolved to determine the motor array. After learning, animats are examined in terms of performance, behaviour and internal structure.

When animats learn to approach a solitary circle, of randomly varying contrast, rectified multiscale coding animats learn to outperform multiscale and intensity coding animats in both independent and coarse scale noise conditions. Analysis of the learned internal structure shows that rectified multiscale filtering facilitates learning by enabling detection of the circle at scales least affected by noise.

Cartwright and Collett (1983) showed that honeybees learn the angle subtended by a featureless landmark to guide movement to a food source at a fixed distance from the landmark, and furthermore, when tested with only the edges of the landmark, still search in the same location. In a simulation of this experiment, animats are reinforced for moving to where the angle subtended by a solitary circle falls within a certain range. Rectified multiscale filtering leads to better performing animats, with fewer hidden units, in both independent and coarse scale visual noise conditions, though for different reasons in each case. Only those animats with rectified multiscale filtering, that learn in the presence of coarse scale noise, show similar generalisation to the honeybees.

Collett, Cartwright and Smith (1986) trained gerbils to search at locations relative to arrangements of landmarks and tested their search patterns in modifications of the training arrangements. These experiments are simulated with landmark distance coded as either a 1-D intensity array, or a 2-D vector array, plus a simple compass sense. Vector coding animats significantly outperform those using intensity coding and do so with fewer hidden units. Furthermore, vector coding animats show a close match to gerbil behaviour in tests with modified landmark arrangements.

Contents

Chapter 1

Introduction

Few animals are fortunate enough to live within easy reach of all their physical requirements for successful life. Each animal has a unique location in physical space, and its resources, such as food, a mate, or its home, which the animal must move in order to exploit the resource. Guiding such directed movement is a major use of animal senses and nervous systems (Dusenbery, 1992).

This thesis investigates a range of visually guided, spatial goal directed tasks using simulated animats embedded within a 2-D environment. The animats are equipped with a 1-D visual array and stochastic control of their direction of movement. Neural computations within animats reactively transform the visual array to a topography preserving motor array that stochastically determines the direction of movement. An individual animats' internal neural network is initially random and hence so is the animats behaviour. Over time, the network, and hence the animat's behaviour adapts in response to a reinforcement signal received only within the spatial goal region.

Firstly, some illustrations of animal spatial goal directed movement are described. These provide evidence of the ubiquity of such behaviour, and motivate the specific tasks simulated later in the thesis. Next, the consequences of embedding animats within environments are discussed, and the particular methodology used in this thesis outlined. Having established the tasks to be studied and the method of study, the biological and computational motivations for convolution as a basis for animat processing are outlined. Finally, the specific aims of the research are outlined.

1.1 Spatial goal directed movement in animals

Mobility frees an organism from the vicissitudes of a sedentary life—a non-moving organism is dead if the resources it requires for life do not happen to be located within reach. Mobility allows an organism the possibility of moving to where its resources are, rather than passively waiting for the resources to come to it. Unguided movement, however, is little better than sitting still, with

the difference decreasing as the resource becomes more sparsely distributed. Therefore, organisms that can move more efficiently than their competitors to an essential resource will have a selective advantage.

Without prior knowledge of the location of the goal, the organism can only sample the spatiotemporal energy pattern of the various physical particles impinging upon its spatial location in order to gain information to guide movement. The selective advantage of an animal that can steer more efficiently than its competitors to a shared but sparse resource led Walter (1953) and Moravec (1988), amongst many others, to argue that the evolutionary pressure this applies has been a primary factor in the evolution of both animal sense organs, and the nervous systems required to extract from them information about distal stimuli required for the animal to efficiently steer toward its spatial goals. A supportive example of this argument is provided by the phylum mollusca. The 50,000 known species of this phylum include snails, slugs, clams, oysters and mussels; sluggish or sedentary invertebrates with primitive sense organs and simple nervous systems. In contrast, one class, the cephalopods, are predators, and have evolved a complex single lens eyes, convergent with the vertebrate eye in many respects, and a complex nervous system capable of advanced learning, and controlling the fast and accurate movement required for success at their predatory lifestyle (Wells, 1962; Campbell, 1993).

The following subsections describe some particular spatial goal directed animal movements. The movement of bacteria is outlined because, simple as they are, they illustrate the essential aspects of spatial goal directed movement that are elaborated by more complex organisms. Limulus is an ancient animal that uses vision to guide a very specific goal directed movement. Insects use their vision for a range of tasks and are able to learn to move to particular spatial locations.

1.1.1 Simple steering

Bacteria are amongst the simplest and oldest of known natural life and are ancestral to all multicellular organisms. Single celled and asexually reproducing, they absorb nutrients through their cell wall. In an environment in which nutrients are unevenly distributed, any bacteria that can move up nutrient concentration gradients will typically intake more nutrients than fellow bacteria that either stay still or just move randomly. Nutrients enhance reproduction and hence those bacteria that can move to where they are concentrated will typically have a selective advantage. About half of all bacterial species are capable of directed movement, the most common mechanism of which is by means of rotating flagella. When the flagella are rotated in one direction, they spiral round one another, propelling the bacterium forward in a roughly straight path called a run. When the flagella are rotated in the opposite direction, the flagella separate, causing an uncoordinated movement, called tumbling, that randomly alters the orientation of the bacterium (Campbell, 1993).

Specific chemical receptor molecules on the cell wall are modified when they come into contact with their target chemical. Either the receptor molecule, or molecules within the bacteria, are sensitive to the change in chemical concentration over short intervals of time. If the chemical is an attractant, and the concentration change positive, then the relative number of tumbles is reduced, leading to runs in the direction of positive gradient. If the concentration change of the attractant is negative, then the relative frequency of tumbles is increased, resulting in random re-orientations until a direction in which the change is positive is found. The net result of this mechanism is a stochastic spatial movement toward higher concentrations of nutrients. In the case of repellent chemicals, the behavioural responses are reversed with respect to the temporal change in the concentration of the chemical. Similar mechanisms underlie the positive phototropism of those mobile bacteria that photosynthesize sunlight (Berg, 1993; Campbell, 1993).

Computationally, this is an ingenious strategy given just one sensor. The temporal derivative of concentration is the same as the spatial derivative in the direction in which the bacteria are heading, by virtue of the fact that bacteria translatory movement is in roughly straight lines.

The spatial goal directed movement of bacteria, though effective in the microscopic world at getting them to their goals, is of very limited general usefulness. For example, a predatory animal whose spatial goal is bacteria, would not be able to locate them by using the same strategy as bacteria use to steer toward their spatial goals. Chemical concentrations are too specific, too short range, and because of diffusion, have little spatial structure at the scale of bacteria.

1.1.2 Basic visually guided steering

In contrast to chemical concentration, light provides an energy source with many useful properties as a sensory messenger. The earth is continually bathed in light, which is both fast, far travelling, and is strongly influenced in spatial and spectral structure by interaction with matter. Most importantly it retains its spatial structure through space: light impinging the sensory surface from neighbouring directions will typically have come from neighbouring points. This enables sensitivity to more subtle variation than just the intensity of light impinging upon the animals body.

The Limulus (horseshoe crab) is a very ancient animal which has remained virtually unchanged for 350 million years. Its eyes comprise about 1000 ommatidia, which vary considerably in sensitivity in a circadian rhythm. Limuli do not use their visual system for food location, or to facilitate any general tasks. Rather it exists to serve just two purposes. Firstly, for predator evasion: limulus's move away in response to large objects looming above them. This is a vestigial behaviour as their predators are long extinct. The only other use of the visual system is to guide movement towards mates (Barlow et al, 1985, Barlow, 1990). This involves detecting, amongst the other elements in the visual environment, the particular spatiotemporal pattern that corresponds

to another Limulus, orienting and then moving toward it (Barlow et al, 1985, Barlow, 1990). Barlow et al (1985) ethologically investigated this behaviour by examining the tendency of Limuli to move toward objects deviating from other Limuli in form and contrast, and by forced choice comparison of Limulus preferences. Their results showed that Limuli approach objects of widely varying form with a preference for negative contrast (Limuli are darker than the sand they reside in); finer discrimination depends upon the tactile sense. Barlow et al (1985) conclude that Limuli move toward bounded patches of negative contrast moving across their ommatidia array.

In chapter 4, a simplified analogy to the task facing Limuli is simulated, with animats learning to approach a solitary circle in an otherwise empty arena. The contrast of the image of the circle varies randomly, and noise is added to the visual array. In order to efficiently perform this task, animats must learn to behaviourally distinguish between variation in the visual array due to the image of the circle, and variation due to visual noise.

1.1.3 Adaptive visually guided steering

The Limulus has a genetically pre-specified behavioural response to its environment, which is sufficient for its lifestyle. Many animals however, including insects and mammals, are able to learn to move to where their resources happen to be. For example, honeybees learn where in their environment flowers are, and use their vision to guide their return to these locations.

Srinivasan (1994) reviews the capacity of honeybees to learn to discriminate between visual patterns. In the typical experimental setup, bees move freely within a Y shaped room in which a spatial pattern is fixed to each of the two arms of the Y. Between trials, the two patterns are randomly swapped between the arms, but food is placed near one of the patterns regardless of which side it is on. Over time, bees learn to discriminate between patterns based purely upon visual cues. The most important information is acquired after learning, when bees are tested on variations of the patterns in order to discriminate between models of what the animals have learned. Bees are able to learn to discriminate between horizontal and vertical stripes; between stripes oriented at -45 degrees and 45 degrees to the vertical (Srinivasan et al, 1993); between patterns differing in either local or global cues (Zhang et al, 1992); and between symmetric and asymmetric patterns (Giurfa et al, 1996).

Cartwright and Collett (1983) showed that honeybees, and Collett, Cartwright and Smith (1986) showed that gerbils are able to learn to move to a spatial location defined only in terms of the visual appearance of an array of landmarks from that position. Having learned to get to the goal, the landmark arrangement is then varied These experiments are discussed in detail and analogous tasks simulated in chapters 4–7.

1.2 Embedment within an environment

A clear prerequisite for spatial goal directed movement is a spatial environment within which the agent has a location. The environment together with the agents sensory equipment determines the sensory state at that location. The sensory state is processed within the agent, resulting in modulation of the agents steering. Any movement leads to a new location, and hence a new sensory state which modulates the movement, which leads to a new location and a new sensory state and so on. Movement depends on sensation and sensation depends on movement. The closed loop nature of such sensorimotor processing characterises the situation in the animal spatial goal directed movement outlined above.

In terms of modeling animals, embedment increases the ecologically validity of the model. Animals have evolved as species, and learn as individuals, to perform their everyday tasks within their environment as well as, or better than, the competition. These form whole behaviours, and their computational basis is the object of computational neuroethological study.

Embedded models can be judged directly, according to how well their behaviour matches that of the animal they model. Embedment removes the interpretive step between the model's input and output and its translation into sensation and behaviour. This behavioural approach to the study of embedded agents frees the research from having to make assumptions about the goal of agent computation between sensation and action. Without embedment, it must be assumed that computation aims to maximise information transfer, or make some feature explicit, or minimize an error function across the stimuli set, or some other non-behavioural criteria. The relation between computation and the behaviour it facilitates is via unimplemented assumption. With autonomous adaptation, the goal of agent computation is solely to produce behaviour suited to the current location given the current task. The computational input *is* sensation, and the computational output *is* behaviour. Animats can thus be judged, as animals are, purely in terms of the effectiveness of their behaviour.

From an engineering perspective, the research goal is the development of autonomous robots that can survive in the hurly-burly of the real world. Such robots, like animals and thermostats, must compute robustly in order to behave appropriately over a wide range of conditions. Thus, Brooks (1986) argues that robots are most effectively developed and studied whilst embedded in the environment in which they must survive.

Given an animat embedded within an environment, the problem becomes finding the computations intervening between sensory input and motor output in order that the resultant animat's behaviour competently performs some task. The approach used in this thesis is to embed adaptive animats within environments and let them autonomously find a computational solution through incremental adaptation. Before discussing this and its implications, the alternative of principled

human design, is discussed.

1.3 Computational analysis

Computational analysis (Marr, 1977) of an isolated task involves determining the computational demands of a task, and from consideration of these, the development of a specification of "what is computed and why" for an agent to perform the task at some level of competence. The result of computational analysis is called the computational theory of the task and, in the best scenario, is a deep understanding of the necessary and sufficient computations underlying competence at the task. Particulars can be deduced from the general computational theory and implemented in an animat. This allows behavioural tests of the computational theory in the extent to which the performance and behaviour of the animat matches that predicted by the computational theory.

When the method works, in the sense of delivering a simply formulated, but general computational theory of a task, computational analysis leads to both an understanding of the task, and a set of machines able to solve it. In this happy case, the task is labeled type 1 by Marr, referring to one end of a continuum of tasks, ordered according to the simplicity of the computational theory that solves them. Computational theories are assessed by two criteria: the simplicity of their formulation – of two computational theories of a task, the simpler (more type 1) is to be preferred; and the generality of their explanatory power – the greater the range of tasks a computational theory can account for (in terms of implementations that behave so as to solve the task), the more useful the theory. Moving along from the type 1 to type 2 end of the continuum, the computational theories increase in complexity, and hence in the number of parameters required to specify them, but decrease in generality in that the size of the set of possible implementations of the computational theory decreases.

Marr (1977) cites the principal difficulty with computational analysis as occurring when a type 1 theory cannot be found for a task. In this case, it is not in general possible to decide if this is a true reflection of a type 2 task, or if the task has a type 1 solution but it just hasn't been found yet. This problem emphasises the human design element of computational analysis — progress awaits the inventiveness of human analysts. Of course, this is somewhat of a caricature of what really happens for the sake of formality. In practice, in all but the simplest of tasks, there will be a more iterative approach to the generation of the computational theory, with examination of the behaviour of implementations of draft theories leading to greater understanding of the computational demands of the task and hence a more accurate computational theory.

Computational analysis has proved very difficult, and this is compounded when the task to be analysed involves an agent embedded within an environment. This is chiefly due to the very wide range of possible computational solutions in all but the simplest of tasks. There is also

no guarantee that a solution proposed via computational analysis is incrementally acquirable, an essential requirement of animal solutions whether through evolution or individual learning. It may be the case that the computational strategy adopted by an animal to solve a task is not the most elegant or theoretically complete, but the most easily evolved or learned.

The most important problem for computational analysis is that each task must be analysed individually, and solution depends upon the ingenuity of human analysers. Autonomous adaptation, in contrast, promises the *automatic* generation of animat behaviour by general systems that become specific through autonomous adaptation in response to the demands of particular tasks.

1.4 Autonomous adaption

An alternative approach to generating animats that can competently perform a task is to specify an environment and embed adaptive autonomous agents within it. Animats are controlled by parameterised computations that modulate behaviour in response to sensory input. The parameters of these computations originally have random values, but are modified over time in response to non-specific feedback evaluating the animats behaviour. For example, in this thesis, the only non-sensory, evaluatory feedback from the environment to the animat is a binary reinforcement signal, which equals 1 if the animat is within the goal region and zero everywhere else. Like much current work, animats here are controlled by neural networks with adaptive weights. Two types of algorithms have emerged for modifying the weights in response to non-specific evaluation of behaviour: genetic algorithms, and reinforcement learning.

Genetic Algorithms

Genetic algorithms (Holland, 1975; Goldberg, 1989) are based on Darwinian principles of evolution by selection of heritable variation. A population of random animats is generated, and the behaviour of each is evaluated on the task with a non-specific fitness measure. A new generation of animats is then generated from the first using genetic operators such as mutation and crossover, with selection biased according to the fitness of each individual. Over a number of generations animats evolve to higher fitness scores, and hence behaviour better suited to the task. The utility of genetic algorithms for evolving successful animat behaviour has been shown by a number of researchers, including Floreano and Mondada (1994).

Reinforcement learning

Reinforcement learning algorithms (eg. Sutton, 1988) modify the weights of neural networks controlling individual animats in response to a delayed, non-specific scalar reinforcement signal. In the temporal difference algorithm used in this thesis, developed by Sutton (1988) and Watkins (1989),

networks learn to map their sensory array to an estimate of how far from the spatial goal they are located given that sensory array. These estimates are utilised for behaviour, as animats should move in the direction leading to sensory states with higher estimates than the current one (and hence nearer to the goal). Learning involves updating the estimates in response to experience of sensory states and actions, and their effectiveness in leading to rapid movement to the goal location. Learning is on-going and depends upon the difference between temporally successive estimates. The temporal difference algorithm used in this thesis is explained in detail in chapter 3.

The end result, after evolution, or reinforcement learning, is an animat that performs the task at some level of competence. If the adapted animat performs well at the task, then the method results in a non-designed machine that behaves competently within its environment. Whatever solution it has developed, it must be incrementally acquirable.

The next step is to try, if possible, to understand the learned computations underlying the animats behaviour. Working out what the animats have learned to do involves examination of both behaviour, and the internal structure of the animat after learning. If animats prove comprehensible then the approach has led to the automatic generation of an incrementally acquirable computational model of the behaviour. If an animat proves incomprehensible, then this is problematic.

1.4.1 Comprehensibility

If an animat is comprehensible, then this means that it is possible to abstract what it is doing in more general computational terms than by specifying the particular animat. For example, in chapter 4, animats learn to approach an object by convolving the sensory array with a step shaped filter. Describing the shape of the learned weight structure is a great deal simpler and more general than a description in terms of the value of each weight. It also implies a more general understanding of what animats have learned in which a range of particular animats can be seen as learned implementations of more general computations. Comparison of animats, in terms of the computations they have learned to implement becomes possible, and hence evaluation of the consistency of computational solutions to a task.

The general computations are abstracted from particular animats, but their accuracy can be determined deductively by implementing particular animats that fall within the set of computations they specify. In the simple case above, step filter animats are hand constructed by setting particular values for the parameters that the abstraction leaves free. The performance and behaviour of the implementation, together with its variation can be empirically compared with those predicted by the computational abstraction. The computational analysis is shifted from the task itself to animats that have learned to efficiently perform the task. Computational analysis of learned animats is analogous to the task facing neuroethologists in computational principles of

animal behaviour. It is at this level of computations underlying the

When an animat proves incomprehensible, this is both uninformative and problematic. As discussed above, Marr (1977) argues that when the computational analysis does not lead to a computational theory, it is problematic because this may be inherent in the task, or due to the poverty of the analysis, and it is not typically possible to decide which. The situation is worse when the analysis is of particular learned animats. Incomprehensibility in this case may be due to the nature of the task, the human analysis of the animats, or due to the algorithm used to generate the animat.

1.4.2 Relationship to animal research

The relation of animats to animals is through comparison of behaviour. In the behavioural experiments examined in this thesis, animals learn, over a number of trials, to locate a food source, by using their vision to guide movement. The most informative aspects of such experiments are trials, after successful learning, in which the environment is modified in some way from the environment during learning. The search pattern in such generalisation trials provides information about what the animals have learned that enables them to perform competently at the learned task, as well as enabling models of the behaviour to be distinguished.

Animats that learn to perform the same tasks as the animals can be posited as models of the behaviour. The test of such an animat model is the extent to which its behaviour matches the behaviour of the animal when tested on modifications of the learning environment. Thus, animats learn within one environment, but are evaluated as models of the animals behaviour in a modified environment, that they have never experienced, and are not permitted to learn from. This is equivalent to human constructed models being only considered worthwhile if they are able to predict more than the data that they were explicitly constructed to model.

1.5 Visual environments

The visual sense of animals is highly complex, reflecting the complexity of the spatiotemporal patterns of light they must extract information from, and involves considerable processing even at the level of photoreceptors. In this thesis, a great many simplifications are made and so the term "simple visual sense" is used to emphasise the gulf between the simulations here and real animal vision, both in terms of the physical properties of light and its sensory reception.

The intensity of light impinging upon a particular photoreceptor of an animal is determined by many factors. These include the position and output of light sources, both primary and secondary; the distance, orientation and reflectance of the surfaces imaged by the receptor, with respect to both the receptor and light sources; and other factors, such as properties of the eye,

and intrinsic and external noise. The resultant activation of the receptor is often itself a complex function of the intensity of light impinging upon it and other variables, such as the activity of neighbouring receptors and variables internal to the animal. Eyes are not merely passive recipients of information, but are dynamically controlled in order to increase the information received. An example is provided by the Limulus, which varies the sensitivity of its ommatidia by a factor of one million in a circadian rhythm, in order to compensate for the daily variation of light in its environment (Barlow, 1990).

In contrast to the 3 dimensional and cluttered world that most animals inhabit, the environments of these simulations are two dimensional and sparse. Surfaces have a scalar reflectance between 0 and 1. There are no light sources, just an ambient illumination that randomly varies between trials. The intensity of simulated light impinging upon a particular receptor depends only on the reflectance of the surfaces it is imaging, and the level of ambient illumination. Neither the orientation nor distance of the surface affect this intensity. One important aspect of the real visual world carried over to the simulation is the variation in mean illumination. Due to the rotation of the earth, the light falling on a location varies by a huge amount on a daily basis. The result is that the absolute level of light intensity falling on a photoreceptor tends to be uninformative, it is the spatial variation in contrast that carries the information. A successful visual system must be able to cope with this variation, and that of most animals does (Barlow and Mollon, 1982). The simulations in this thesis reflect this aspect of the world by randomly varying the simulated ambient illumination, so that only intensity differences, and not absolute intensities, carry any useful information. A further aspect carried over from the real visual world to the simulations in this thesis is visual noise, both independent and coarse scale.

1.6 Convolution

Convolution is a standard mathematical operation for transforming one array to another of the same dimensionality and size (Bracewell, 1965). A linear filter is independently centered at each position in the input array, and the value of the corresponding position in the output array determined as the sum of input array values around that position, weighted by the filter. Hence, the pattern of values in the output array depends upon both the pattern of input array values and the shape of the filter. More generally, and in this thesis, convolution may be with a non-linear filter, and employ subsampling. The essential idea remains the same: repeated application of an identical filter for a topography preserving transformation from one array to another. Convolution has found very widespread application in many areas of science (Bracewell, 1965), and characterises some of the processing in the early stages of much animal vision.

The compound eye of an insect such as a honey bee or housefly consists of around 3,000 facet

lenses, each focusing on a set of eight or nine photoreceptors. The early visual system of insects consists of three successive ganglia: the lamina, medulla, and lobula. Connections between these ganglia are highly ordered and genetically prespecified with great precision (Osorio et al 1994).

Photoreceptor output projects in an orderly one to one mapping to large monopolar cells (LMC) in the lamina, which transform the receptor array so as to use the coding range of neurons as efficiently as possible (Laughlin, 1987). In the projection from lamina to medulla, topography is again preserved. Medulla neurons are organised into orderly columns, with one column corresponding to each LMC in the lamina and each column containing around 50 neurons. Unlike the receptor array to lamina projection, where each LMC receives input form a single receptor neuron, medulla neurons typically integrate over a localised region of activity centered on the LMC in the topographically equivalent position to the medulla neuron (Strausfeld, 1976). Medulla neurons are precisely ordered: within each column, the 50 neurons have a wide range of receptive field structures, with neurons sensitive to both spatial and temporal aspects of stimuli falling within their receptive field. In contrast to the diversity within each column, across columns the response properties of each neuron are highly consistent. For each receptive field location, there is a neuron with the same receptive field structure in each column across the medulla, and hence the eye of the insect. Furthermore, across individuals, medullas are highly consistent, with the same set of 50 receptive field structures in each column (Osorio et al, 1994).

Thee lamina to medulla transformation in insects can be characterised as convolution of the lamina array by 50 filters of diverse spatiotemporal shape. The result of this is 50 arrays of the same size as the laminal array — a hugely expansive recoding of the sensory array.

The same principle of convolution by a range of filters holds in the transformation from medulla to lobula, this time with subsampling (one lobula column for every 4 medulla columns), but involving a similar number of filters (Osorio et al, 1994). However, in the medulla to lobula transformation, convolution is not over a single 2-dimensional array (as in the receptor to lamina and lamina to medulla transformation), but over 50 2-dimensional arrays, each having spatiotemporal patterns of activity reflecting different aspects of the visual array.

The early visual system of mammals embodies similar computational principles. Retinal ganglion cells convolve the retinal array with filters having center surround structure, and filters with a spatiotemporal structure (Bruce and Green, 1985). As first discovered by Hubel and Wiesel (1968), the early areas of visual cortex convolve the array of LGN activations by a diverse range of orientation, scale, colour, and movement sensitive filters (Bruce and Green, 1985). Like the insect optic lobe, mammalian early visual cortex is highly ordered, with neurons organised into columns, and later area filters convolving over the many arrays resulting from earlier convolutions. Additionally like insects, the transformations are hugely expansive with many filters in each layer.

Tanaka et al (1996) argue, from neuropsychological study, that the convolution architecture

extends at least as far as inferotemporal cortex, with neurons there able to respond to very complex aspects of the visual array by convolving across the arrays resulting from numerous previous layers of convolution.

The similarity in the processing performed by the medulla and lobula of insects, and the early areas of visual cortex in mammals has been noted by a number of researchers (eg. Srinvivasan, 1994; O'Carroll, 1993; Srinivasan et al, 1993). Similar, convolution based architectures are also found in the retina and optic tectum of amphibians (Ewert, 1984; Ewert, 1987; Young, 1989). It must be emphasised that in addition to the feedforward, convolution architecture emphasised here, the early visual systems of animals involve extensive feedback and lateral connections.

Convolution of the visual receptor array by a number of filters of diverse spatiotemporal structure, followed by further convolution of the arrays of filter output, is a partial characterisation of the early visual systems of animals as diverse as insects and mammals. The shape of the filter may be determined mostly genetically, as in insects, or be strongly modulated by learning within the individual, as in mammal cortex. Early auditory processing is also based on convolution of a sensory surface consisting of an ordered array of frequency receptors (Gallistel, 1990). In contrast, the early olfactory areas of both insects and mammals do not utilise convolution, which Osorio et al (1994) argue reflects the lack of spatial structure in the olfactory receptor array.

1.6.1 Convolution networks

Convolution by adaptive, non-linear filters, implemented as neural networks with a single output unit, is the computation of animats within this thesis. One, or more, convolution layers transform the 1-D visual array into a 1-D, topography preserving motor array that stochastically determines direction.

The architecture is prespecified in the sense that the receptor array, and the convolution architecture is fixed. The number and size of filters is prespecified, as is the degree of subsampling and the nature of the non-linear integration. Reinforcement learning modifies the weights of the adaptive filter networks.

The supervised learning of convolution networks weights was developed by Rumelhart et al (1986) and involves straightforward modifications of the standard back-propagation algorithm. They have been used for handwritten character recognition, though here the convolution tends to be by very small filters responding to the minutiae of the input images. A review of the application of supervised convolution networks is provided by LeCun and Bengio (1995). Fukushima (1989) uses an unsupervised learning convolution network for handwritten character recognition, but as in Le Cun's architecture, the first stage filters are very small compared to the size of image features such as letters or words.

In addition to the biological motivation for using convolution networks, there are a number

of computational advantages of this architecture. Translation invariance is built in, rather than having to be learned. It is presumably the lack of a preferred horizontal orientation for animals that elicits convolution by the same filter across the visual field. In the vertical plane, for insects, there is a preferred orientation, and this is reflected in the differences in filter shape in the vertical plane. Just as for animals, there is no bearing preference for the animats in this research and so building in translation invariance by using a convolution architecture might be supposed useful in that it builds a computational property into the architecture, rather than it having to be learned.

For animals, the bearing of images in the horizontal plane tends to be unconstrained, and so it would seem useful to be able to respond identically. In the case of the image of an object, the pattern of activation specifies aspects of the object, and the relative location of the pattern specifies the objects bearing.

The genetically prespecified convolutions of the early visual system of insects reflects this fact about images. In addition to the computational utility of convolution, the repetition of the same structure allows a more efficient genetic encoding and hence more rapid adaption through natural selection. For mammals, the convolutions in the early visual areas of the cortex are largely learned, suggesting that mammals individually come to reflect this computational demand through individual plasticity, for the same reasons as insects have evolved the structure.

The computational utility of convolution networks is achieved with a computational economy in that, because of the repetition of the weights, they have far less parameters than fully connected networks of the same size. Additionally, it can be expected that with fewer weights, learning is faster (Haykin, 1994). As will be demonstrated in later chapters, this also makes analysing what the animats have learned easier because behaviour can often be inferred from the response profile of individual learned filters.

1.6.2 Multiscale filtering

As described above, the early visual system of diverse animals can be partially characterised as an implementation of repeated convolution by a range of filters; the animats in this thesis share this structure. However, in addition to sharing the computation of convolution, evidence is accumulating that diverse animals are convolving with the same shaped filters. In particular, a subset of filters in the medulla and lobula of insects have both a scale and orientation selectivity, a feature of filters in the mammalian cortex.

The existence of multiple scale and orientation sensitive filters has been suggested in insects by both behavioural (eg. Zhang et al, 1992; Srinivasan et al, 1993; Srinvivasan, 1994), and neurophysiological methods (eg. O'Carroll, 1993). Srinivasan et al (1993) found that insects could learn to discriminate between patterns of random stripes differing only in orientation. Zhang et al (1992) found that insects can learn to discriminate between patterns differing only in local or

global cues independently and make use of whichever is available and useful in a particular task. They suggest that this behaviour is mediated by channels sensitive to different ranges of spatial frequency. O'Carroll (1993) mapped the receptive field of orientation and scale sensitive filters within the lobula of dragonflies.

Psychophysical evidence for the existence of multiple scale and orientation sensitive filtering within the early cortical areas of the mammalian visual system is widespread and reviewed by Bruce and Green (1985) and Watt (1988). Physiological evidence was provided by Hubel and Wiesel (1968).

The question therefore is what is the computational utility of such filtering? The computational analysis approach to answering the question of why animals have particular shaped filters is based upon consideration of the properties of the filter and supposition of what their role may be in behaviour (eg Marr, 1982; Watt 1988). Autonomous adaption allows a different approach to this problem based upon assessing a particular set of filters according to how well animats, having their visual array convolved by such filters, can learn to perform a task. Thus, the utility of particular filters is first established behaviourally. Their computational role can then be assessed by examining the behaviour and internal structure of animats using them.

In this thesis, the visual array is 1-D and so orientation is not an issue; the utility of multi-scale filtering of the visual array is examined by comparing the performance of animats having a multiscale filtered visual array, with animats having an unfiltered array. If those with multiscale filtered visual arrays learn to perform at a higher level than those without, this suggests that such filtering facilitates learning. By comparing over a range of tasks, it becomes possible to specify the conditions in which filtering facilitates learning and by analysing animats after learning examine why.

1.7 Reactive agents

A reactive agent is one whose behaviour at any time depends solely upon its sensory input at that time. It has no information about the temporal variation in either sensory state or its internal variables. A reactive task is one solvable by a reactive agent.

The animats in this thesis are reactive. The visual array is transformed by a feedforward neural network to the motor array, used to stochastically determine the direction of movement. Animats have no direct memory of either the visual array or their movements on previous time steps. Nor do these animats have internal variables that depend on recent behaviour as provided by recurrent networks (eg Cliff et al, 1997).

Reactive animats were chosen for simplicity of processing and to facilitate analysis of behaviour and internal structure after learning. With reactive animats, a single motor array is associated

with each location in the environment. With non-reactive animats behaviour depends on both where the animat is, and where it has come from.

Whilst there is a great deal of evidence of non-reactive processing in animals, they is also a reactive component to their processing. Bees can learn to differentiate between stationary patterns: Srinivasan et al (1993) found bees in a Y shape environment could learn to distinguish between horizontal and vertical stripes presented for only 0.2ms every 0.5 sec; too quickly for motion cues. Cartwright and Collett (1983) found bees learned the visual angle subtended by a featureless landmark to guide search; this is a reactive cue available in stationary images.

Thus, animals are sensitive to both stationary retinal images and temporal variation in the images. The reactive animats of this thesis demonstrate what can be done in these tasks with just spatial filters. Sensitivity to image motion seems partially meditated by filters with a temporal as well as spatial structure within a convolution architecture. Extension of the work to non-reactive animats with such filters is discussed in chapter 8.

1.7.1 Why not just do supervised learning?

Given the reactive nature of the animats, this raises the objection, why not just do supervised learning of the required input-output mapping? The animat could be placed at a random location in the environment, and the sensory input at that location mapped via supervised learning to the direction that leads the animat nearest to the goal. This method should result in animats that move efficiently to the goal, since the required function is likely to be learnable given enough hidden units.

There are two major reasons why this method is not good. Firstly, animats must learn to move to the goal based upon the reinforcement signal received only once at the goal. Clearly this is, in general, a much harder task than learning the supervised mapping. Animals do not have an external teacher pointing to the goal at each time step; they must work out for themselves how to get there from personal experience. Reinforcement learning animats face the same problem and so reinforcement learning of the movement has an ecological validity that supervised learning does not.

The second reason why the supervised learning of these tasks is not useful is that it is not possible in general to uniquely specify what the input–output mapping is. In all but the most trivial of tasks, there are a range of different strategies for effectively moving to the goal. Animats develop particular routes and strategies for goal finding which depend upon factors like network size, sensory coding, and noise. The animats in this thesis, like animals, tend not to learn the most efficient routes to the goal, but the easiest learned routes given their situation. Animats because they are embedded able to some extent control their sensory input. Actions in some regions of the environment may not be learned at all, but because these regions are rarely visited given the

animats goal finding strategy, this has little effect upon overall performance. These effects are not available for disembodied mapping.

To give a concrete example of this, the simplest task in this thesis is in chapter 4, where the environment is empty except for a solitary circle which animats must learn to approach. Even in this simplest case, it is found that some animats approach the edge of the circle and some approach it head on. Which of these strategies is learned depends on the animats sensory coding. In the case of landmark learning (chapter 7), where the environment contains numerous landmarks, there are many different strategies for getting to the goal and animats will be shown to often find ingenious strategies for getting there with the minimum of computational effort.

1.8 Aims of the research in this thesis

Throughout the thesis, animats with the same convolution based internal structure learn to perform a range of visually guided, spatial goal directed tasks in simulated 2-D environments. A 1-D visual array is reactively mapped, through convolution, to a topography preserving motor array that stochastically determines the direction of movement. The higher the value of a motor array element, the higher the probability that the animat will move in the corresponding direction.

The same reinforcement learning algorithm modifies the filter networks controlling animat behaviour based upon a binary reinforcement received only at the goal. Thus, the first question is the extent to which this animat design and adaption algorithm can generate animats able to efficiently learn to move to the spatial goal in the particular task.

In chapters 4–6 two forms of coding the visual array are compared: either the visual array itself is convolved by an adaptive filter network to generate the motor array; or the visual is convolved by multiscale filters to yield a 2-D multiscale array which is convolved by the filter network to determine the motor array. The question here is whether multiscale filtering of the visual array leads to animats that learn to perform more efficiently than those convolving the raw visual array, and how this is affected by visual noise and task. The overall goal is to determine, in behavioural terms, the computational utility of multiscale filtering: under what circumstances, and why, does this computation yield better behaviour.

In chapter 4, animats learn to approach a solitary circle in an otherwise empty arena; the contrast between landmark and wall luminance varies randomly between trials, both in sign and magnitude. With no visual noise, animats learn to efficiently approach the circle whether or not their visual array is multiscale filtered. However, when visual noise is present, animats with multiscale filtering outperform those without, and this performance difference increases with noise. Hence it is concluded in this case that multiscale filtering can lessen the deterioration due to visual noise. Analysis of the learned computations reveals the mechanisms underlying this noise

resistance.

Chapter 5 usues the same environment as above, but instead of the goal region being around the circle, it is where the visual angle subtended by the circle falls within a particular range. Even without visual noise, multiscale filtering leads to significantly superior performance. This difference increases in the presence of noise.

In chapter 7, the sensory array codes landmark distance, which is coded either as a 1-D intensity array or expanded to a 2-D vector coded array. Once again, the computational utility of these forms of coding is compared through the behaviour of animats that learn to use them. The behaviour of animats is compared to that of gerbils in Collett et al's (1986) landmark learning experiment. These tasks involve a few landmarks in fixed relation to each other and to an invisible goal region. Analogously to the finding that multiscale filtering facilitates learning the visual tasks of chapters 4 to 6, local coding of the distance array facilitates the learning of these spatial navigation tasks. Further simulations compare the search behaviour of animats to the animals when the landmark arrangement is modified; it is found that vector coding animats behave closer to the gerbils than intensity coding animats.

In all cases, animats that have learned to efficiently move to the spatial goal are animats are examined after learning in terms of performance, behaviour and internal structure. The goal of this analysis is determine what computations the animats have learned that underlies their efficient behaviour.

Chapter 2

Literature review

This chapter reviews some of the computational psychology research most relevant to the research presented in this thesis. The focus is upon animats, whether real or simulated, that use visual information to guide movement. At the end of the chapter, the relationship between the research in this thesis and the reviewed literature is discussed.

Beer (1995, 1996) focuses mostly upon animat locomotion, rather than the visual processing studied here. Arbib (1987) focuses mostly upon high-level visual schemas, rather than the low level vision studied here. Hence, the work of these researchers will not be discussed further.

2.1 Walter's tortoise

W. Grey Walter (1953), designed and built a simple electromechanical animat that he called Machina Speculatrix. This had a single photoelectric sensor, and a touch sensor. Speed and direction of movement were controlled by the animat, which resembled a tortoise. Rather than being a computer program running on a general purpose computer, as is generally the case today, animat control was by a small circuit of thermionic valves, relays and condensers.

Machina Speculatrix was wired so that the photosensor constantly rotated through 360 degrees, until activated by a light source, in which case it stopped rotating. The machine steered towards moderate intensity light, but avoided very bright light. This was implemented via an ingenious design in which the front wheel, which controls the direction of movement, was directly connected to, and pointed in the same direction as the rotating photoreceptor. In darkness the photoreceptor and front wheel rotate continuously, resulting in roughly straight movement. When activated by a light source, the photoreceptor stops rotating, and so does the front wheel, and the animat moves toward the light.

If placed equidistant from two equal light sources, the animat does not move between them;

instead the photosensor will become active in response to one of the lights, stop rotating and steer toward the light. As the machine moves closer to the light, the activation of the photoreceptor will increase, until, if the light is bright enough, the avoidance mechanism will be activated. This behaviour, like the steering toward mechanism, emerges from Walter's physical design. The photoreceptor rotates away from bright light, which also causes the front wheel to rotate in the same direction, which causes movement away from the bright light. As the photoreceptor continues rotating, it will fix upon the less intense other light source and steer toward this. Left alone in this situation, machina speculatrix will continue to move to and fro between the two light sources.

A touch receptor was wired into circuits so that upon hitting an object, the animat would reverse a small distance, rotate the front wheel somewhat, and then move forward. The result is simple but effective and robust obstacle avoidance.

Walter attached lightbulbs to animats, with the result that they became attracted to each other. If the lightbulbs are bright enough, the bright light avoidance behaviour will be activated, and they will only move to within a certain distance of each other, but no closer. If the lightbulbs are not bright enough to elicit the bright light avoidance behaviour, animats will move toward each other until they meet, which activates the touch sensitive obstacle avoidance behaviour, so they back off, become attracted again and so on.

Simple reinforcement learning was also implemented by Walter. An auditory receptor was added to Machina Speculatrix, together with circuitry to keep a slowly decaying trace of the derivative of the activity of the light and sound receptors. Given a binary signal, the derivative of activity is highest at stimulus onset. The animat was prewired to move toward moderate light, and hence Walter regarded this as an unconditioned reflex. If a whistle is blown, followed some time later by the turning on of light, then the animat moves toward the light. If the trace of activity in the sound receptor decays slowly enough, then it will overlap with the trace of activity in the photoreceptor. Given such as overlap, the circuitry of Machina Speculatrix increased the weight between the two activity traces, so that activity of the sound receptor would lead to increase in activity of the valve responding to the photoreceptor. Over time, with repeated pairing of sound and swiftly following light, the weight becomes large enough so that activity of the sound receptor elicits behaviour without the light.

Walter's work demonstrates the relative complexity, and surprisingly animal-like behaviour that can emerge from very simple receptor and internal processing mechanisms. It also illustrates how the constraints of actually building a physical machine, in contrast to the freedom of general purpose computation, may lead to ingeniously simple mechanisms underlying seemingly complex behaviour. Phototaxis is built into, and inseparable from, the actual mechanical structure of the animat. The extent to which the same may be true of animals is an important question.

2.2 Braitenberg's vehicles

Braitenberg (1984) published a series of thought experiments with simple hand-designed animats that he called vehicles. They are simple enough for their general behaviour to be imagined, and were neither simulated or implemented on a real machine. Vehicles with two light sensors at their front end and two wheels at their rear end to control the direction of movement, exhibit a variety of behaviours depending upon the pattern of connection between the sensors and the wheels. When each of the sensors is connected to the wheel on the opposite side of the vehicle, it will move toward a light source. When the sensors are connected to the wheel on the same side of the vehicle, the vehicle moves away from light sources. Like Walter (1953), Braitenberg examines the behaviour of multiple vehicles with lights attached to them and explores the ensuing dances of attraction and repulsion.

Braitenberg progresses through more complex vehicles within the same framework to explore behaviours involving primitive learning, simple pattern detection and movement detection.

In contrast to the specific physical mechanisms that underlie the behaviour of Walter's animat, Braitenberg's work abstracts the essence of sensorimotor problems, and develops very general strategies for solving them.

2.3 Brooks' subsumption architecture

Brooks (1986,1991) has been highly influential in the development of this field, which he emphasises as a behaviour based approach in contrast to the knowledge and representation based approach of traditional artificial intelligence research.

Brooks's work has focused mostly upon architectures for control of robots. Rather than have a central and general behavioral controller, Brooks decomposes complex behaviour into a collection of simple and specific task achieving behaviours. Each of these is performed by an autonomous behavioural module dedicated to a particular task. Higher levels of control are achieved by modules which process the output of lower level modules, and can suppress, or subsume, them. The result is a robust and flexible robot control system.

2.4 Modelling insect visuomotor control

Fransceschini, Pichon and Blanes (1992) develop a robot visual system explicitly modelled on aspects of the visual system of houseflies. The system was implemented on a real robot using purpose built parallel circuitry. The robot had 100 photoreceptors, arranged in a horizontal plane, nonlinearly covering 360 degrees. The number and arrangement of receptors is close to that for a horizontal slice through the receptor array of a housefly. This 1-dimensional array of

intensity values is convolved by an array of elementary movement detectors (EMD) modelling the computations of neurons identified in the lobula of houseflies. EMD's compute the difference between the activity in one region of an array, and the activity in a neighbouring region after a short temporal delay. They respond most strongly to contours moving at a preferred speed in a preferred direction, and can be thought of as filters with temporal as well as spatial structure. Covering the whole visual field, EMD's transform the receptor array to an array in which high activity at a point signals movement in the direction, and at the velocity at which the EMD's are sensitive. By convolving the image with a number of EMD's, each with a different preferred direction, sensitivity to all directions can be obtained. In this case, the image is transformed into a set of images, one for each direction, and the evidence suggests that this is the sort of organisation employed by many species of both vertebrates and invertebrates (reviewed by Franscechini et al, 1992).

Franscechini et al's (1992) animat convolves with two sets EMD's, one for each direction in the 1-D visual array. The task for the animat is to move to a goal location, the bearing of which is provided as additional input to the animat, while avoiding obstacles. The obstacles cause points of high contrast in the 1-dimensional visual array; movement of the animat results in movement of the contrast points, and this elicits activity in the EMD layer. Because the image is one dimensional, movement can only be in one of two directions and hence two sets of EMD's are needed, with opposite preferred directions. By the simple principle of motion parallax, a moving agent can compute the relative distance of a contrast point from its angular speed as it moves across the visual field due to the agents movement (Whiteside and Samuel, 1970). Franscechini et al (1992), use the activity in the prewired, EMD layer as input to a motion parallax calculation that outputs the relative distance of contrast points and hence obstacles. This together with the bearing of the goal is used to steer the animat so that it moves toward the goal whilst avoiding obstacles. The animat performed well at the task, being able to slalom through a cluttered environment toward the goal at speeds of around 50 cm per second.

2.5 Reinforcement learning animats

Prescott and Mayhew (1992) report a neural network controlled, simulated animat, that learned to avoid obstacles. The sensory input to their animat is provided by three range finder sensors. This three dimensional state-space of possible sensory values is split into non-overlapping boxes to produce a locally coded sensory array.

The animat moved around in a cluttered environment and received a negative reinforcement signal upon bumping into an obstacle. Obstacle avoidance alone is not, in general, a well formed task, because animats can just stay still, or move in a tight circle, and they will avoid obstacles.

Animats whose goal is solely obstacle avoidance do indeed learn such strategies. In order to avoid such trivial, but effective, strategies, Prescott and Mayhew (1992) introduced the further constraint that the animat, whose speed was constant, should make as few turns as possible. This was incorporated in the reinforcement regime and resulted in animats that both competently avoided obstacles, and covered a lot of ground.

2.6 Evolving animat controllers

As discussed in chapter 1, genetic algorithms, based on Darwin's theory of biological evolution by the selection of heritable variation, provide a method of autonomously generating animat behaviours in response to non-specific feedback evaluating those behaviours. Genetic algorithms have proved highly effective in evolving behaviours of neural network controlled animats.

Floreano and Mondada (1994) evolved neural networks for a mobile robots having a 1-D, 8 element, sensory array of proximity detectors. Animats were evaluated according to how well they avoided obstacles whilst maximising forward velocity in a simple maze-like environment. A population of 80 animats evolved effective solutions to this problem within 100 generations; the best evolved animats were shown to be considerably more efficient than hand-wired animats based on Braitenberg's (1984) vehicles.

Floreano and Mondada (1996a) used the same mobile robots and environment to explore emergent homing behaviours. Animats were equipped with a short-life simulated battery, and a location in the environment, specified by visual cues, was designated the battery recharge area. The fitness function used to evaluate animats was a simplified version of that used by Floreano and Mondada (1994) and did not contain any explicit terms for driving the animat toward the recharging area or for processing the sensory cues associated with it. Animats that lived longer however were able to score higher evaluations and this implicit pressure led to complex and highly interesting emergent behaviours. Animats would efficiently move around the environment when their battery levels were high, thus scoring high evaluations. When their batteries were low, however, they would engage in the completely different behaviour of detecting and steering toward the battery recharge area. Recharging their batteries enabled them to continue with the efficient movement behaviour and thus further increase their evaluation. Floreano and Mondada (1996a) emphasise that evaluation functions should be as general and simple as possible, and like the situation with animals, only implicitly pressure animats to engage in particular behaviours.

In an extension of this work, Floreano and Mondada (1996b) evolve the weight change rules determining the learning of neural networks controlling the animats. Individual synapses between units in the network had a genetically coded weight change rule. Animats then learned to perform the "go fast whilst avoiding obstacles task" of Floreano and Mondada (1994), starting from random

weights. Evolution provided the animats with learning rules leading to rapid development of efficient behaviour.

Cliff, Harvey and Husbands use genetic algorithms to adapt the parameters of neural network controlled animats in order to evolve a range of visually guided behaviours (Cliff et al, 1993; Harvey et al, 1994; Cliff et al, 1997). The animats are implemented both in a real robot, with a 2-dimensional camera, and in an accurate simulation of the robot. The research goal is the evolution of animats that can competently perform tasks of incrementally increasing complexity. They use a very general animat design based upon variable size, continuous time recurrent neural networks controlling animats whose visual systems are also parameterised and subject to evolution. Their design explicitly builds in as little structure as possible. A genetic algorithm, developed by Harvey (Harvey, 1991, 1993), modifies all these parameters in response to the selection pressure of task competence. The size of the networks, weights between units, and visual receptor arrangement are all modified by this algorithm. Thus, very general animats become specialised though evolution to solve particular tasks.

Cliff et al (1997) argue that animals, and hence the animats that model them, are most usefully viewed as continuous dynamical systems. Within this perspective, sensory or other input to the system acts to perturb the trajectory of the dynamic system in state space. The perturbation may lead to changes in the variables determining motor control, and hence to a change in behaviour. A perturbation, however, may not directly influence behaviour, but instead lead to changes in internal variables which may indirectly alter behaviour at a later time. The most important point is that Cliff et al explicitly reject a view of sensorimotor processing as a reactive transformation from a visual array to a motor array. Cliff et al's animat design reflects their dynamical system view of behaviour, by being controlled by a recurrent neural network.

The parameters defining the photoreceptors are themselves under genetic control and evolve concurrently with the neural network they provide input to. The simulated, or real, camera provides a 2-dimensional, high resolution visual image which is sampled by photoreceptors, defined by two parameters, a center, and radius of their receptive fields. Each photoreceptor averages the activity within its receptive field, and this provides the visual input to the animats.

In one of their simplest tasks (reviewed in Cliff et al, 1997), animats are placed in random locations within an empty circular arena and must move to the center of the arena as quickly as possible, and stay there. The evaluation function that selects for competence at this task sums the distance of the animat from the center of the arena over time. Thus, the more time a particular animat spends at the center of the arena, the lower its evaluation function. The genes defining animats with the lowest evaluations in a generation have the greatest probability of being retained to define animats of the next generation. Over 100 generations, with a population of 60 animats, the genetic algorithm evolves animats that perform very well at this task. The best animats move

directly to the center of the arena from any start position and then rotate in a tight circle around the center.

Analysing animats that have evolved to perform a task efficiently forms a major part of Cliff et al's research, and involves identifying attractors in the state space of animat-environment interaction. These attractors correspond to stable relationships between the animat and its environment, such as the animat being in the center of the arena. In this task, two photoreceptors were used in the evolved solutions, and two different solutions to the task were evolved which, though leading to similar behaviour, involve different internal structures and positions of the two photoreceptors.

Harvey et al (1994) report the evolution of animats able to discriminate between a circle and square within the environment by using an evaluation function which over time sums the animats distance to one and subtracts the distance from the other. Efficient solutions were evolved that used two photoreceptors and a comprehensible internal structure.

2.7 Relation of the research of this thesis to the literature

This section attempts to place the present research in the context of the work outlined above, and to justify the various decisions made here concerning animat design and research focus.

2.7.1 Processing

Like much current research, this research uses adaptive neural networks to control animats. Cliff et al (1997) use a very general, recurrent network architecture, in which both the network structure, and the weights evolve in response to the selection pressure of the evaluation function. In the present research, the network architecture is fixed in that the only computation is convolution. In the multiscale coding condition, the visual array is convolved by a number of filters with fixed structure; in all conditions the visual array, or the multiscale filtered array, is convolved by a neural network whose weights change over time, in accordance with the reinforcement learning algorithm.

As explained in chapter 1, animat processing here is based on the computation of convolution, because this is widely found in the early stages of both vertebrate and invertebrate vision. This approach follows Francesini et al (1992), who as outlined above, take convolution as the basic computational building block of their insect modelling animat design. Francesini et al convolve with hand wired filters in order to perform a particular set of behaviours. In the present research, the convolution is by an adaptive neural network which learns a structure in response to the computational demands of the task.

2.7.2 Adaptation

Of the research discussed above, Walters (1953), Braitenberg (1984), and Francescini et al (1992) used predominantly hard wired networks. Cliff et al, and Floreano et al, and much of the current research in autonomous agents use genetic algorithms to evolve populations of animats in response to the selection pressure of evaluation functions.

Here, reinforcement learning is used to adapt the weights of individual learning animats. The reinforcement regime is the same throughout this research: animats receive zero reinforcement on each time step except when they move within the goal region, when a reinforcement signal of 1.0 is received. This setup is close to the animal behaviour experiments (such as Cartwright and Collett, 1983; Collett and Cartwright, 1986) that are the focus of the present research, in which individual animals are let loose in an environment containing an invisible food source. The animals begin having no knowledge as to where the food is, and are only rewarded with food when they get to the right location. In such situations, animals are effective at individually learning to visually guide their movement to the location of the food.

2.7.3 Multiple task learning

This research uses the same animat design for a range of tasks in which only environment and goal location differ. General animats learn to become specialised in response to the particular environment and spatial goal that define a particular task.

2.7.4 Comprehensibility

Following work above such as Floreano and Mondada (1994, 1996a, 1996b), a major focus of this research is on analysing the behaviour and learned internal structure of animats that have learned to perform a task competently.

The convolution based animat design used here facilitates analysis in a number of ways. Because the processing is reactive and feedforward, behaviour at a location in the environment is determined stochastically by the visual input at that location, and is unaffected by the route the animat took to get there. Furthermore, because animats have 360 degree field of view, and the behaviour at each time step is to choose a direction rather than alter the direction of movement, there is no orientation parameter. At each location, a single sensory array is transformed into a single motor array, which specifies the probability that the animat will move in each direction.

Although clearly straying from the situation with animals, these simplifications greatly facilitate analysis of what animats have learned to compute.

2.7.5 Relation to animal behaviour research

This type of work yields computations that produce behaviours. These computations have not been hand-designed by a human programmer, but have been autonomously generated adaptation of individual animats in response to the demands of the task.

In the case of this thesis, in chapters 5 and 7, animats learn tasks that are analogous to those that animals have been experimentally shown to be capable of learning. In chapter 5, the analogous animal behaviour experiments involve honeybees learning to move to a goal location at a fixed distance from a single cylindrical landmark (Cartwright and Collett, 1983). In chapter 7, the experiments involve gerbils learning to move to a goal location defined relative to a number of cylindrical landmarks (Collett et al, 1986). These two sets of animal experiments are ideal material for the present thesis for a number of reasons.

In both sets of experiments after learning the task, animals were tested in a variety of modifications of the environment during learning. It is these generalisation tests that are crucial in discriminating between models. Two animats that differ in some way may both learn to both competently perform a particular task, but if the difference between them is significant, then they will behave differently in some modified environment. This situation is found repeatedly in later chapters, where it is shown that although animats behave identically when tested with the environment they learned in, they behave differently, but consistently, when tested with modifications of the learning environment. Thus, behavioural experiments of this kind provide computational psychologists with both a behaviour to model, and a clear means for assessing the accuracy, and hence refuting, proposed models.

These tasks are good for the present purposes because there are no explicit models of the behaviour, in the sense of a well defined set of computations that, when implemented, result in behaviour matching the animal's in the learned environment *and* in the range of modified environments tested by the experimenter. The method of computational analysis has (to date) failed to yield models of these behaviours and so they are prime targets for modelling by incremental adaptation.

Chapter 3

Reinforcement learning

In reinforcement learning tasks, a learning agent produces a sequence of actions and receives a delayed scalar reinforcement signal which indicates the effectiveness of the actions. However, the reinforcement received does not indicate which actions were responsible for its value, nor is it informative about the way actions should be modified to increase reinforcement. For example, in this thesis, reinforcement is zero at all spatial locations except within the goal region, where a reinforcement signal of one is received. Having randomly stumbled upon the goal region, how can the animat use the non-specific reinforcement signal, received only having arrived at the goal, to modify its actions so as to steer more efficiently toward it in the following trials?

Temporal difference learning (Sutton, 1988) is a method for solving this credit assignment problem. Firstly, the basic temporal difference learning algorithm is explained. This provably solves the reinforcement learning problem if it can be framed as a Markov decision problem. Next, the extension of this algorithm to less restrictive and more general cases using neural networks as adaptive function approximators is discussed. Finally, the application of the temporal difference algorithm, Q learning (Watkins, 1989), to the convolution animats used here is explained. This section specifies the learning algorithm used throughout this thesis.

3.1 Temporal difference learning

Consider an agent, which at each time step is in one of a finite set of states X, and does one action from a finite set of actions, A. At time t ($t = 0, 1, 2...$), the agent is in state $x_t \in X$ and performs action $a_t \in A$. The result is that the agent deterministically transitions to state $x_{t+1} \in X$, independently of its past history. When the state transition occurs, the agent receives reinforcement, r_t, a function of x_t and a_t. The choice of action in each state is known as the policy of the agent.

The goal of the agent is to maximise the cumulative reinforcement received over time, which amounts to moving, as soon as possible, to those states where reinforcement is high, whilst avoiding states where reinforcement is low. However, reinforcement may be sparse, and states of low reinforcement may be the only route to higher reinforcement, and so it is not generally optimal to just move to the state with highest immediate reinforcement.

Firstly, suppose the agent has a policy for determining which actions to perform in each state, and consider the problem of estimating the expected cumulative reinforcement to be received in the future given the current policy. Define the evaluation of state x_t, as the discounted cumulutive reinforcement to be received in the future, starting from time t:

$$V(x_t) = \sum_{k=0}^{\infty} \gamma^k r_{t+k} \tag{3.1}$$

$0 \leq \gamma \leq 1$ is a discount factor which weights reinforcement according to how far in the future it is received. If $\gamma = 0$, then $V(x_t) = r_t$ and is maximised by performing the action that will lead to the highest immediate reinforcement. As γ gets closer to 1, longer term reinforcement becomes more important. In the single goal case of this thesis, where $r_t = 0$ except when the agent moves to the goal state, where $r_t = 1$, $V(x_t)$ reduces to γ^λ, where λ is the number of steps between state x_t and the goal state. Thus, the evaluation of a state is a measure of how far away it is from the goal state, given the current policy.

Temporal difference algorithms rely on a recursive rewriting of the above evaluation function that is made possible by the exponential definition of discounted reinforcement

$$\begin{aligned} V(x_t) &= \sum_{k=0}^{\infty} \gamma^k r_{t+k} \\ &= r_t + \sum_{k=1}^{\infty} \gamma^k r_{t+k} \\ &= r_t + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+1+k} \\ &= r_t + \gamma V(x_{t+1}) \end{aligned} \tag{3.2}$$

That is, the evaluation of the current state equals the immediate reinforcement plus the evaluation of the next state, discounted by γ. It is this recursive definition of state evaluation that provides the key to temporal difference learning algorithms. In each state, the agent estimates $V(x_t)$, by a function, $V'(x)$. $V'(x)$, will be an accurate estimate of $V(x_t)$ when two temporally succesive estimates $V'(x_t)$ and $V'(x_{t+1})$ satisy equation 3.2. Hence the estimates can be adjusted by using the difference between the two sides of the equation as an error on the estimate of the evaluation of state x_t

$$error(V'(x_t)) = r_t + \gamma V'(x_{t+1}) - V'(x_t)$$

$V'(x_t)$ can then be incrementally updated in the direction that reduces this error. Barto et al (1989) show that this algorithm converges to an accurate estimate of the evaluation of each state. That is, $V'(x) = V(x)$ for all $x \in X$.

Given a policy, the above algorithm provides a way of iterating toward an accurate evaluation of that policy. The next step is to see the link between an estimate of the evaluation of each state, and a policy for action selection. Given an evaluation function, the best policy is to do the action, at each time step, leading to the state with the highest evaluation. Thus, given a policy, an accurate evaluation function can be incrementally learned, and given an evaluation of each state, optimal actions can be generated.

Q learning, developed by Watkins (1989), exploits this close relationship between evaluation and policy. Instead of estimating the evaluation of states, the utility of state/action pairs is estimated. Define the utility of doing action a_t when in state x_t, $Q(x_t, a_t)$, as the immediate reinforcement plus the discounted evaluation of the next state

$$Q(x_t, a_t) = r_t + \gamma V(x_{t+1}) \tag{3.3}$$

Assuming that the policy of the agent in each state is to do the action leading to the state with highest Q value, then (Watkins, 1989)

$$Q(x_t, a_t) = r_t + \gamma \max_{a \in A}(Q(x_{t+1}, a_{t+1})) \tag{3.4}$$

This link between temporally succesive Q values allows an incremental algorithm for their estimation to be developed. Let $Q'(x_t, a_t)$ be the agents estimate of the utility of doing action a_t in state x_t. In each state, the agent does one of the actions, leading to a new state, x_{t+1}. Then, the difference between estimates of the utility of actions in the new state, and the preditions from the previous state provide an error on that estimate

$$error(Q'(x_t, a_t)) = r_t + \gamma \max_{a \in A}(Q'(x_{t+1}, a_{t+1})) - Q'(x_t, a_t) \tag{3.5}$$

$Q'(x_t, a_t)$ can then be incrementally updated in the direction that reduces this error. Note that only the estimate for the action actually perfermed in state x_t is updated, since from the single experience nothing is known of the utilities of other actions. A consequence of this is that although the best action in each state is the one with highest Q value, other actions should be tried in order to converge upon accurate estimates for all actions in each state. Stochastic action selection, whereby actions are chosen randomly, but with a higher probability according to their

Q values, is thus required. This is refered to as exploration, as opposed to the exploitation of doing the action in each state with the highest Q value (Thrun, 1991). Watkins (1989) shows that this algorithm converges, so that given enough time for learning, $Q'(x) = Q(x)$ for all $x \in X$ and $a \in A$.

3.2 Q learning with neural networks

The above algorithm can be mathematically shown to converge in the highly restrictive conditions whereby the agent is moving amongst a finite number of distinguishable states. In this case, estimates are single numbers associated with each state. In the more general case, as in this thesis, agents have a sensory array which is a symptom of their state rather than the state itself. Sensory arrays may be continuous, and need not all be distinguishable. Neural networks provide a way of mapping such arrays to utility estimates by virtue of their general function approximation properties. The sensory array is mapped by a neural network to a set of Q value estimates, the errror term calculated as above, and the neural network updated to to reduce the error (Barto, Sutton and Watkins, 1989; Lin, 1992). Whilst this permits reinforcement learning in more general situations than the Markov decision problems discussed above, mathematical proofs of convergence are not avalable. Thus, whether learning in a particular case is possible becomes an empirical question. Lin (1992) and Tesauro (1992), amongst others, have demonstrated the utility of Q learning with neural networks in this more geneal case.

3.3 Q learning for convolution animats

Here, the standard Q learning algorithm described above, is used to update filter network weights in the convolution animat architecture (see fig. 3.1). This section descibes the algorithm, which remains the same throughout the thesis.

3.3.1 Filter networks

Sensory input to the animats is provided by a 1-D visual array. In this thesis, the array evenly covers 360 degrees, though this is not essential to the algorithm. The sole internal processing component of animats is a filter network, a standard feedforward network with a single output unit (fig. 3.1a) with activation between 0 and 1. Direct filter networks have a single layer of weights mediating between the input units and the single output unit, in which case, the activation of the output unit is simply the weighted sum of the input unit activities put through a sigmoid function

$$o = f\left(\sum_i w_i x_i\right)$$

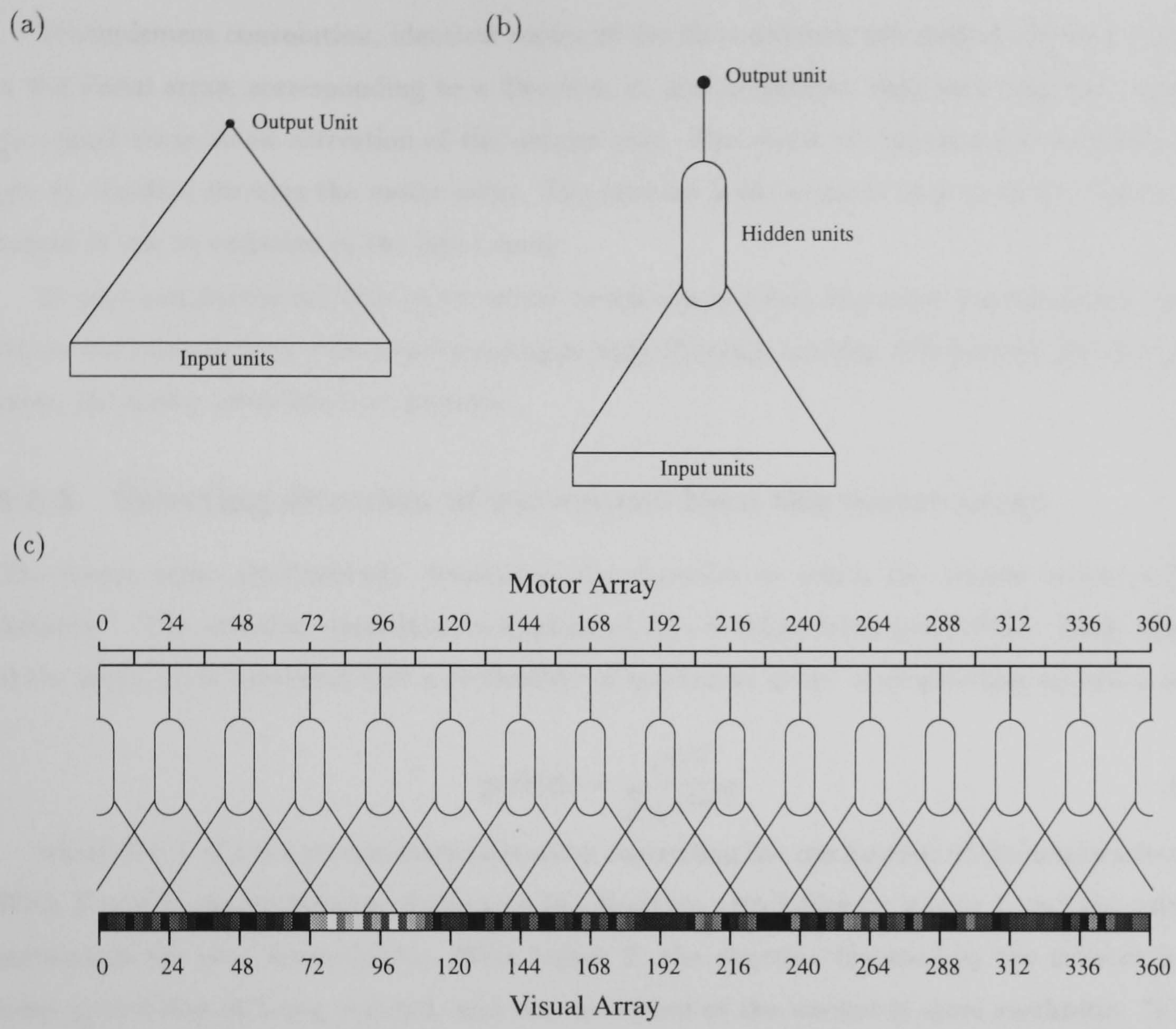


Figure 3.1: Animat sensorimotor system. Filter networks: (a) Direct. (b) with hidden units. (c) The 1-D visual array is convolved with the adaptive filter network to produce a 1-D motor array which stochastically determines the direction of movement.

where o is the output unit activation, x_i is the activity of the ith input unit and w_i the weight to it. The sigmoid function is given by

$$f(x) = \frac{1}{1 + e^{-x}}$$

If the filter network has hidden units, each of these has an activation given by the sigmoid of the weighted sum of input activity. The output unit activity is then given by the sigmoid of weighted sum of hidden unit activation.

3.3.2 Mapping from visual array to motor array

Sensory input to the animats is provided by a 1-D visual array, **x** covering 360 degrees, a function of the animats location within a 2-D environment. This array is convolved with a filter network to produce a 1-D motor array (fig. 3.1).

To implement convolution, identical copies of the filter network are centered at each position in the visual array, corresponding to a direction, θ, and in parallel, they each map the region of the visual array to an activation of the output unit. The result is a number for each direction, $q(\mathbf{x}, \theta)$, together forming the motor array. The network is the same at each position; variation in output is due to variation in the input array.

To save computational time on the serial computers in which this work was simulated, convolution was implemented with a subsampling so that although covering 360 degrees, like the visual array, the motor array has less elements.

3.3.3 Selecting direction of movement from the motor array

The motor array stochastically determines the direction in which the animat moves a fixed distance.[1] The selection algorithm to implement this is taken from Lin (1992). Each element of the array, q_i, is converted into a probability of movement in the corresponding direction, θ_i

$$prob(\theta_i) = \frac{e^{q_i/T}}{\sum_k e^{q_k/T}} \tag{3.6}$$

where $0 < T \leq 1$ is a temperature parameter controlling the randomness of the action selection. With T near 0, the probability of selecting the direction with highest q is near 1, and the animats movements are near deterministic. With higher T, the direction favoured by the network has a lower probability of being selected, and the movement of the animat is more stochastic. In this thesis, and following Lin (1992), the same value of T is used throughout learning and testing of all animats.

A movement direction θ_m is randomly selected given these probabilities, and the animat moves a fixed distance[2] in this direction.

3.3.4 Updating the weights of the filter network

The new location yields a new visual array, $\mathbf{x}'$ which is mapped by the unchanged filter network to a new array of values $q'_i = F(\mathbf{x}', \theta_i)$. The Q learning temporal difference algorithm outlined above is used to generate the error signal with respect to the direction actually moved, θ_m

$$error = \begin{cases} 1 - q_m & \text{if the current location is within the goal region} \\ \gamma \max_k(q'_k) - q_m & \text{otherwise} \end{cases} \tag{3.7}$$

This error is then backpropagated through the filter network so that weights are changed with respect to the input to the network in the direction in which the animat moved, θ_m. Standard

[1] Throughout this thesis, the step size is 10.

[2] Throughout this thesis, the step size is 10.

backpropagation (Rumelhart et al, 1986) with momentum is used throughout this thesis.

If the current location is not within the goal region, the visual array is then convolved with the updated filter network to produce the q values for the current location, and the cycle repeated.

3.3.5 Parameters

Throughout the thesis, the reinforcement learning paramaters are kept constant and follow Lin (1992). Temperature, T (in the stochastic action selection) = 0.02; Discount, $\gamma = 0.95$; backpropagation learning rate = 0.2; backpropagation momentum = 0.9.

Chapter 4

Learning to approach a solitary circle guided by a simple visual sense

4.1 Introduction

In this, and the following two chapters, the environment is a 2-D circular arena, empty except for a solitary circle (see fig. 4.1). In this chapter, the circle is of fixed radius, and the task of the animats is to move, in as few steps as possible, to within a short distance of the circle from any starting location in the arena. For each trial, the simulated illuminance due to the circle and the arena wall are chosen randomly from the range zero to one. These will be referred to as the circle and wall intensities. Animats have evenly spaced photoreceptors, the activity of which is calculated as the mean intensity of surfaces within its response region.

Animat visual input is a 120 element, 1-D visual array of continuous values, evenly covering 360 degrees. Animat motor output is a 1-D array of continuous values, which codes the probability of moving a fixed distance[1] in each of 15 evenly spaced directions. The internal structure of animats consists of a topography preserving mapping between these two arrays involving convolution by fixed and adaptive filters.

Three forms of coding of the visual array are compared: intensity, multiscale, and rectified multiscale coding. With intensity coding, the raw visual array is convolved with a filter network with a single output unit to produce the motor array. With multiscale coding, the visual array is convolved with a number of Laplacian of Gaussian (LoG) filters at a range of scales to produce a

[1] Throughout this thesis, the step size is 10.

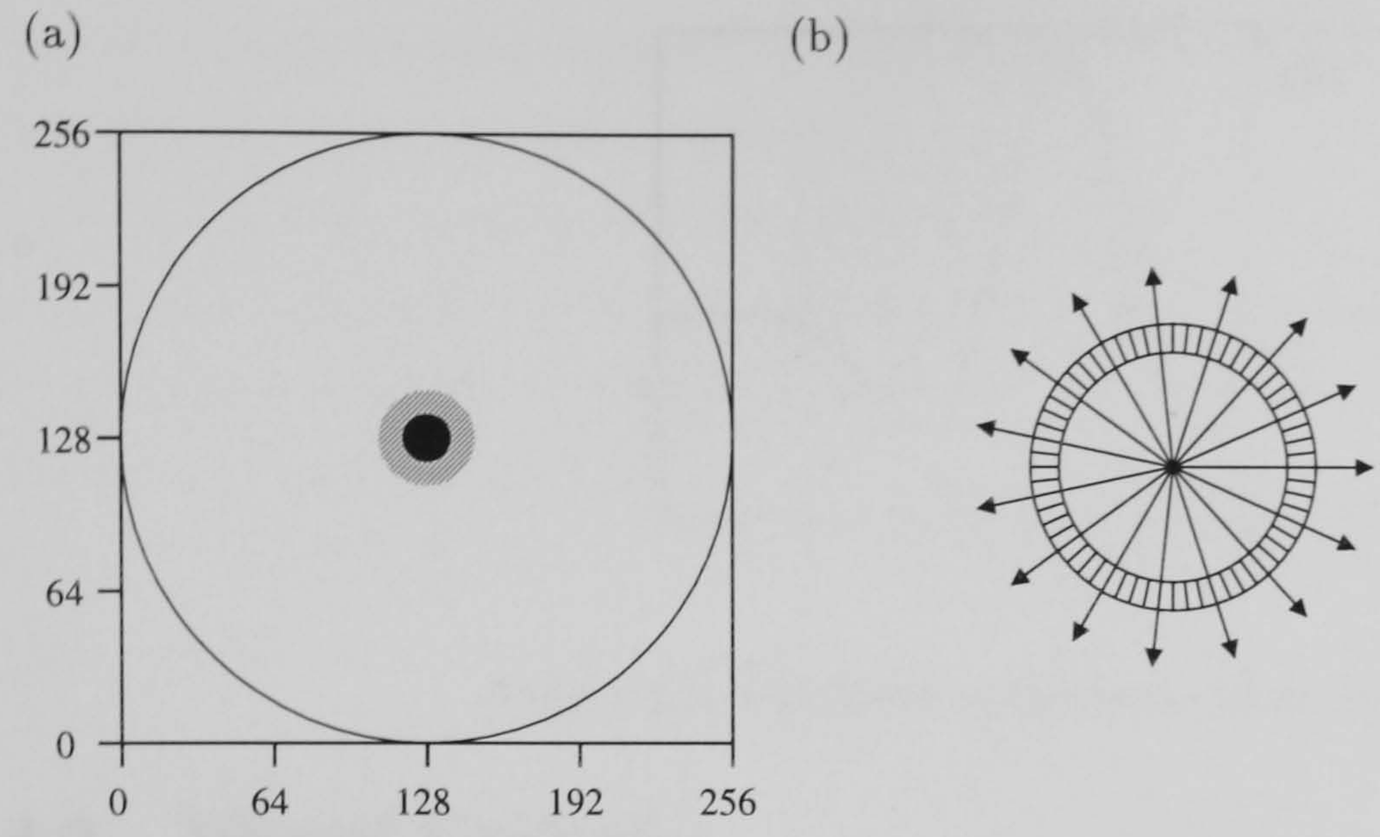


Figure 4.1: (a) The environment is a circular arena of radius 128, empty except for a solitary circle of radius 10. The invisible goal region, shown in gray, is a circle of radius 20, centered on the circle.
(b) Animats have a 1-D visual array, evenly covering 360 degrees. This is mapped to a 1-D motor array that stochastically determines the direction of movement at each time step.

2-D, multiscale array. This 2-D array is then convolved by a filter network to produce the motor array. With rectified multiscale coding, the multiscale array is split into two arrays; one containing the positive elements and the other the negative. These two, 2-D arrays are then convolved with a filter network to produce the motor array.

With the exception of chapter 7, where the visual input is modified, identical animats and coding conditions are used throughout this thesis. The following two sections provide a more detailed account of the visual input, visual coding, and internal processing of animats.

4.1.1 The visual array

In an environment consisting of a solitary featureless circle in a featureless arena and no sensory noise, each element of the visual array has one of two values: either the wall or circle intensity. These are chosen randomly and independently from between 0 and 1 on each trial; the contrast therefore may be either positive or negative and has a magnitude between 0 and 1. The visual array consists of a compact region of elements at the circle intensity, with all other elements at the wall intensity, as shown in fig. 4.3a. The position of the image of the circle within the visual array is a function of its bearing with respect to the animat; the visual angle subtended by the circle is a function of its radius and distance from the animat.

Independent visual noise is simulated by adding a random Gaussian value of mean 0.0 to each element in the visual array. Figs. 4.3b and c show examples for the two independent noise levels used in this chapter: standard deviations of 0.05 and 0.1. Details and figures of coarse scale visual noise are in section 4.7.

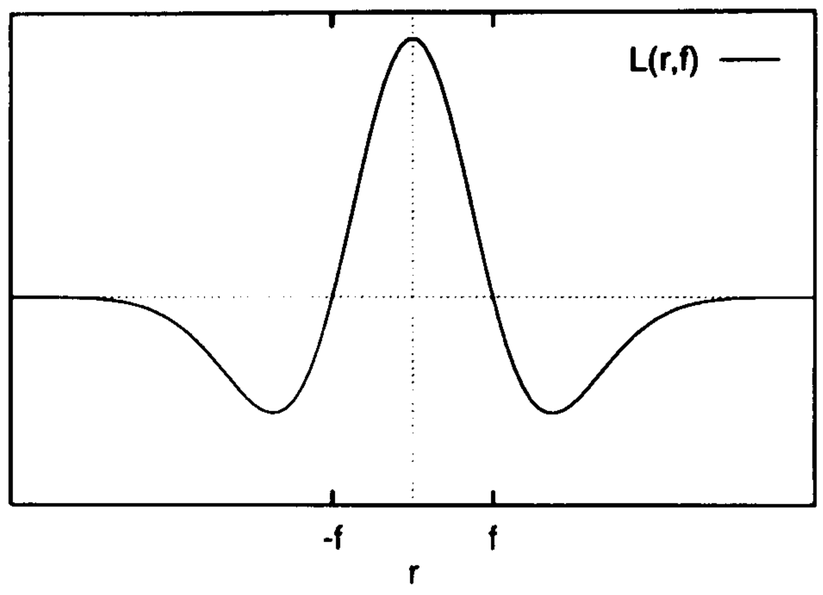


Figure 4.2: Laplacian of Gaussian filter of scale f.

4.2 Visual Coding

4.2.1 Intensity Coding

In the intensity coding condition, the raw visual array is itself convolved with a filter network to determine the motor array.

4.2.2 Multiscale coding

With multiscale coding, the 1-D visual array is convolved with Laplacian of Gaussian (LoG) filters at a range of spatial scales, resulting in a 2-D array. The 1-D LoG function is derived as the negated second differential of a standard Gaussian function with standard deviation, f. The LoG of spatial scale parameter, f, is given by (see fig. 4.2):-

$$L(r,f) = (1 - \frac{r^2}{f^2})e^{-r^2/2f^2}$$

This function has two important mathematical properties. Firstly, the function is balanced: its integral is equal to zero. Hence, when a single valued array is convolved with a LoG the response is zero regardless of that value. LoGs are only sensitive to contrast, rather than the absolute value of spatial patterns. Secondly, they are sensitive to the spatial scale of a pattern, responding maximally to image variation at a particular scale, defined by their parameter, f. Variation at a scale much smaller, or larger, than this produces a near zero response. The LoG function is a 1-D approximation to early visual filters found in a wide range of animals (Marr, 1982; Watt, 1988; Young, 1989).

The integral of LoG(r,f) equals zero. However, the integral in the positive, or negative regions equals $2f/\sqrt{e}$, a function of f. This means that the filter response to a pattern is scaled by a factor dependent on f, whereas the important aspect is the relation between the filter scale and the spatial input pattern. Hence, normalised LoGs are used in this thesis, obtained by dividing L(r,f) by $2f/\sqrt{e}$. In this case, the positive and negative integral of the LoG equals 1 regardless

of scale, and the function obtained by convolving it with a spatial pattern depends only on the *relative* scale of the LoG and the pattern.

Convolving at a number of scales results in a 2-D, multiscale array, with each row being the convolution at a particular scale, as shown in fig. 4.3. The convolution is linear and without subsampling so each row of the multiscale array is the same size as the visual array. Throughout this thesis, the same 6, exponentially increasing, scales are used. For clarity, they are numbered 0 to 5, and the following table lists the corresponding f parameters in terms of elements of the visual array and degrees of visual angle (each element in the visual array has a resolution of 3 degrees):

Scale Number	Visual array units	Degrees of subtended angle
0	1.0	3.0
1	1.6	4.9
2	2.7	8.1
3	4.4	13.3
4	7.3	21.9
5	12.0	36.0

Fig. 4.3 shows the multiscale convolutions of three example visual arrays. The multiscale array is of size 6×120. In the noiseless case (fig. 4.3a), most elements in the multiscale array are zero. Maximum response occurs at a scale dependent on the angle subtended by the circle. Independent noise (fig. 4.3b and c) causes most activity at the finest spatial scales, with activity due to noise decreasing as the scale increases. As can be seen in fig. 4.3b, negative contrast results in a negation of the multiscale array.

4.2.3 Rectified multiscale coding

For rectified multiscale coding, the multiscale array is split into two arrays of the same size as the multiscale array. The positive array is obtained by setting all negative elements of the multiscale array to zero. The negative array is obtained by setting all positive elements to zero, and reversing the sign of the negative elements. The result is two, 2-D arrays of size 6×120 containing values greater than or equal to zero.

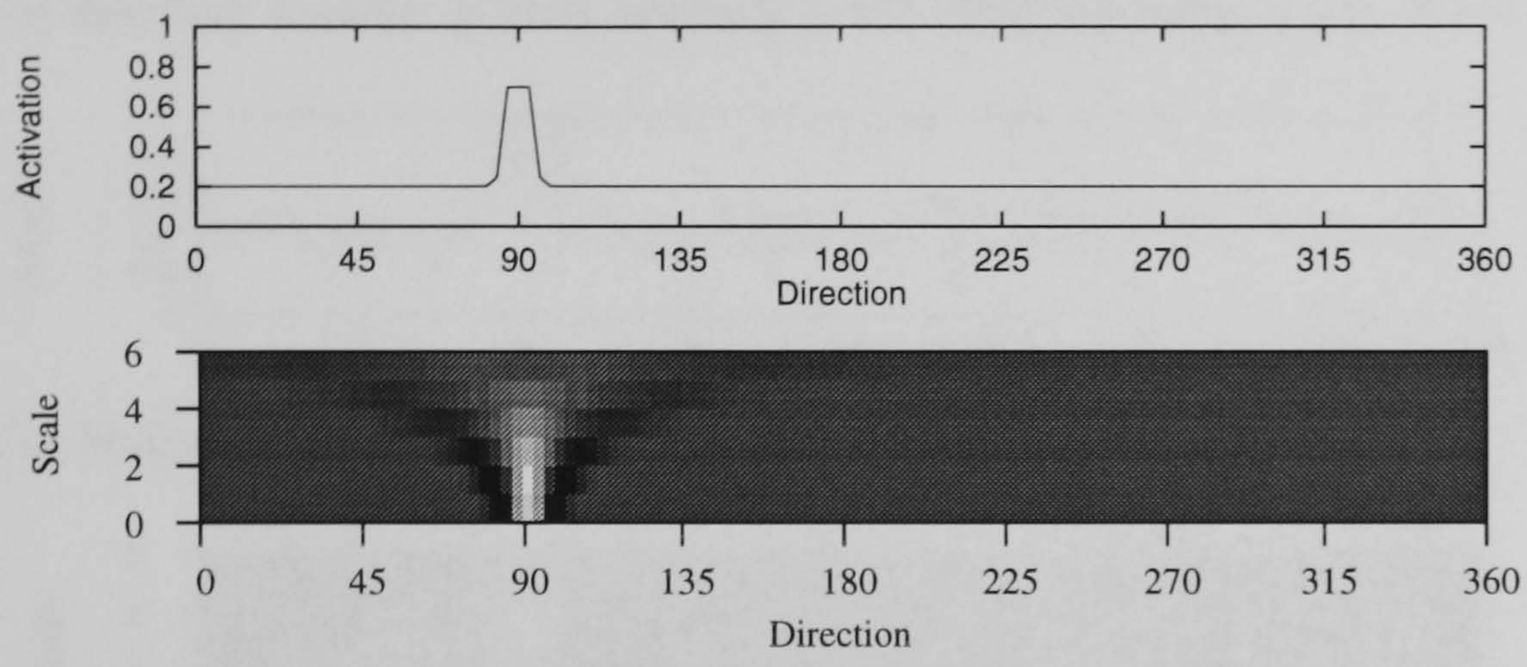
Fig. 4.4 shows the three coding schemes for a typical visual array.

4.3 Animat internal processing

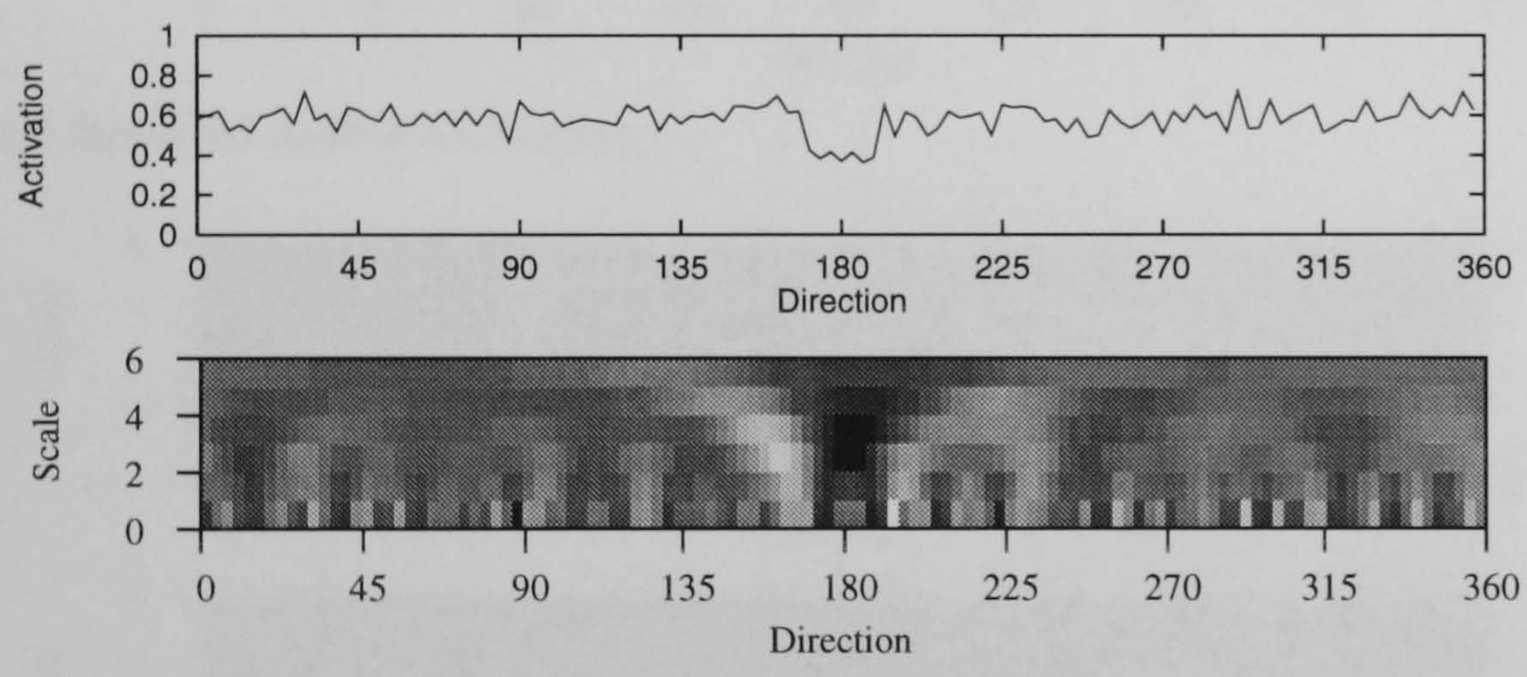
As described above, in the three different visual coding conditions, the visual array is coded as either: a 1-D, 120 element array (intensity coding); a 2-D array of size 6×120 (multiscale coding), or 2 2-D arrays of size 6×120 (rectified multiscale coding).

In all cases, the array is then convolved with a filter network; a standard feedforward neural

(a) Distance = 120. Circle intensity = 0.7. Wall intensity = 0.2. Noise = 0.00



(b) Distance = 60. Circle intensity = 0.4. Wall intensity = 0.6. Noise = 0.05



(c) Distance = 20. Circle intensity = 0.8. Wall intensity = 0.4. Noise = 0.10

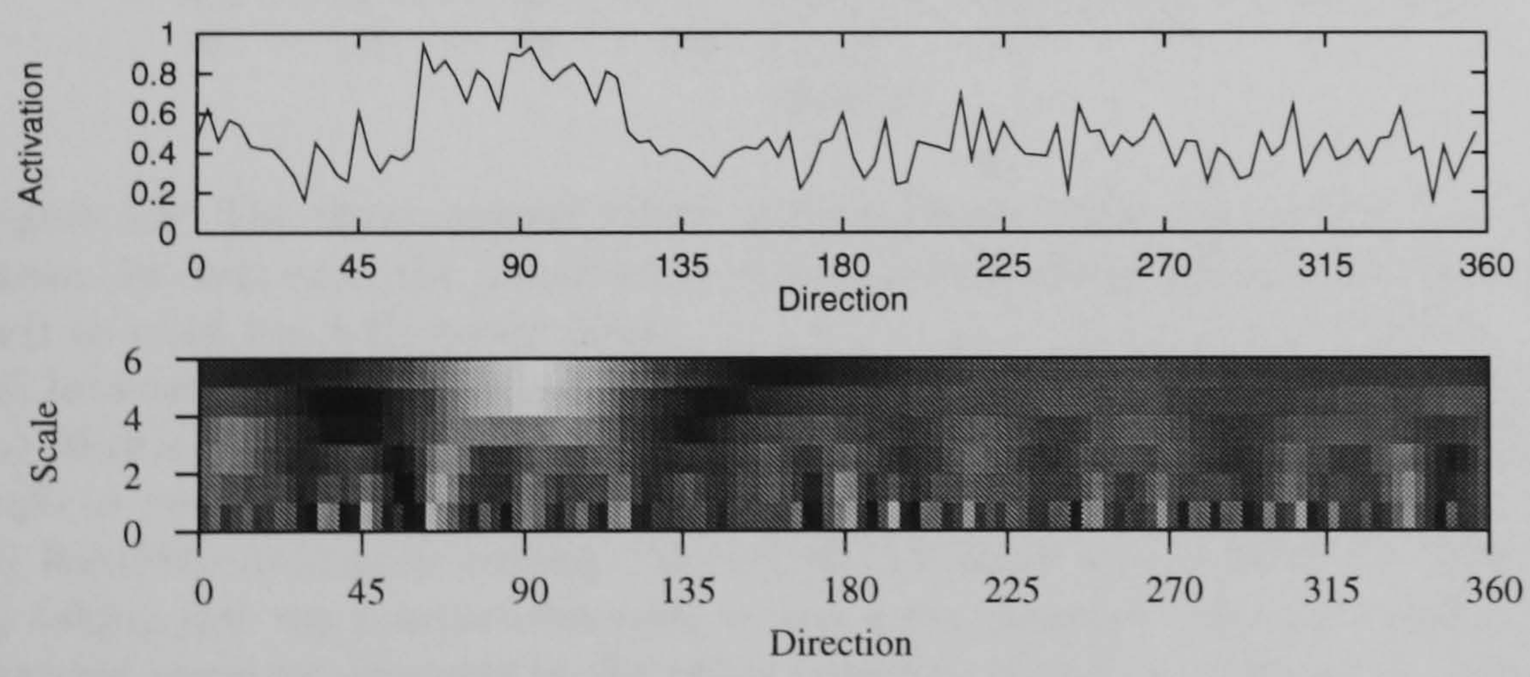
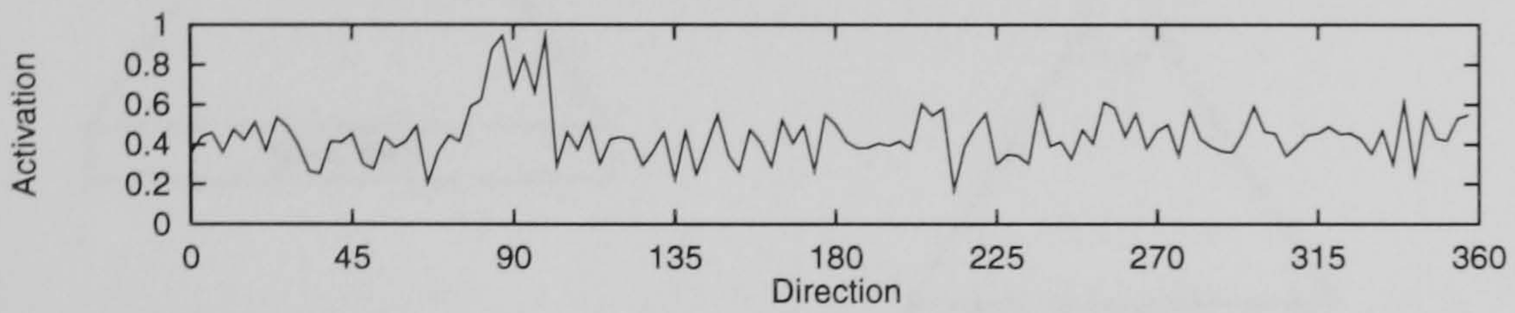
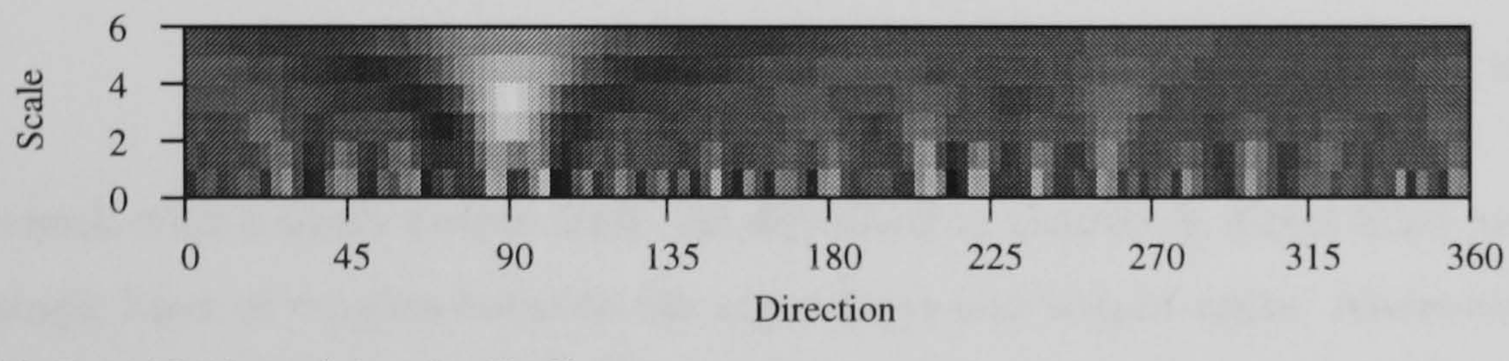


Figure 4.3: Example visual arrays for the current task, and their multiscale convolutions. The 1-D visual arrays (continuous values between 0 and 1) are possible filter network input in the intensity coding condition. The multiscale convolution arrays (positive and negative values) are possible filter network input in the multiscale coding condition.

(a) Intensity Coding. (Circle intensity = 0.7. Wall intensity = 0.4. Noise = 0.10)



(b) Multiscale Coding



(c) Rectified multiscale Coding

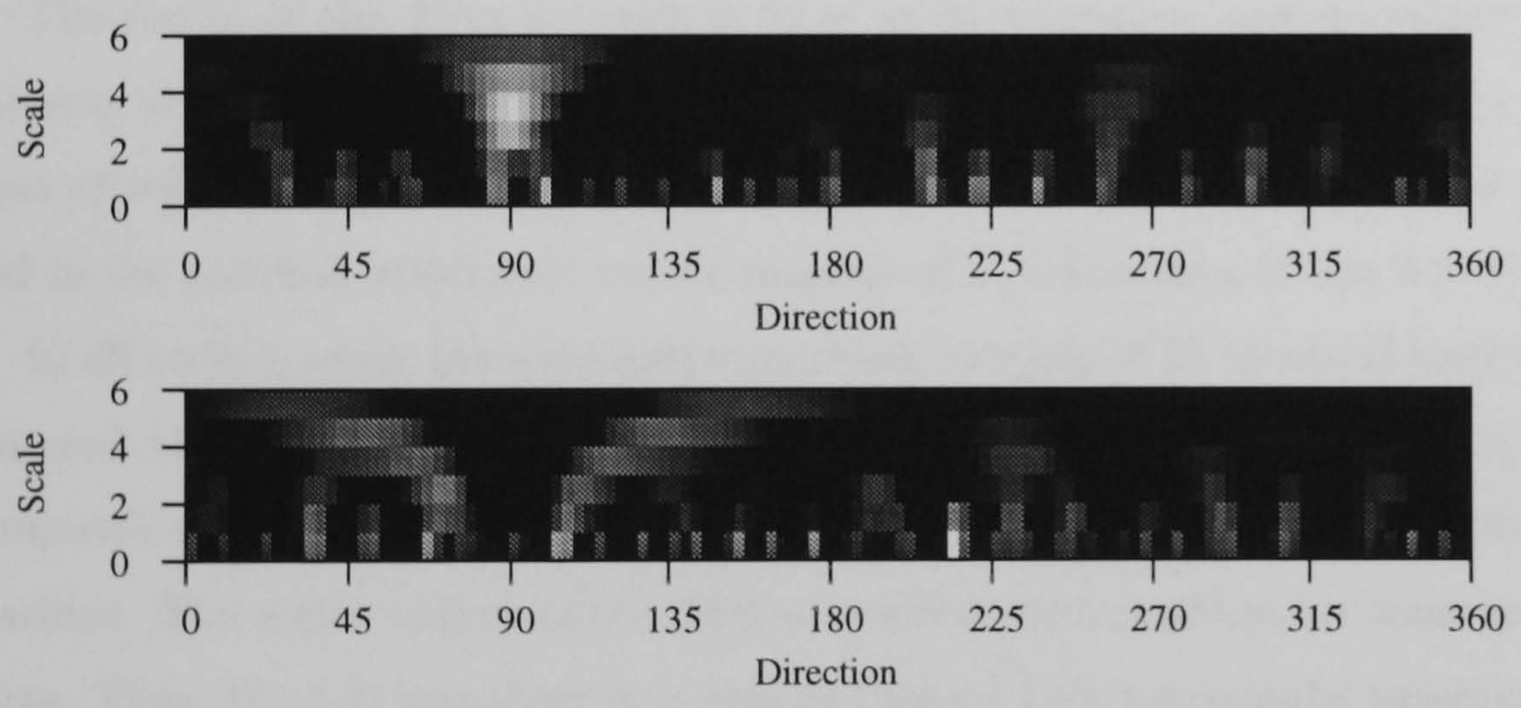


Figure 4.4: The three animat visual coding schemes that are compared in the simulations in this thesis. In each case, the coded array is then convolved with a filter network having a single output unit to yield the 1-D motor array.
(a) Intensity coding : the raw, 1-D visual array of elements between 0 and 1.
(b) Multiscale coding : the visual array convolved by 6 LoG filters of exponentially increasing scale to yield a 2-D array of positive and negative values.
(c) Rectified multiscale coding : the multiscale array is split into two, 2-D arrays of the same size by taking just the positive elements in one array (negative elements set to zero), and just the sign changed negative elements in the other (positive elements set to zero). Yields 2 arrays of positive or zero values.
In each case, the array is then convolved with a filter network having a single output unit to yield the 1-D motor array.

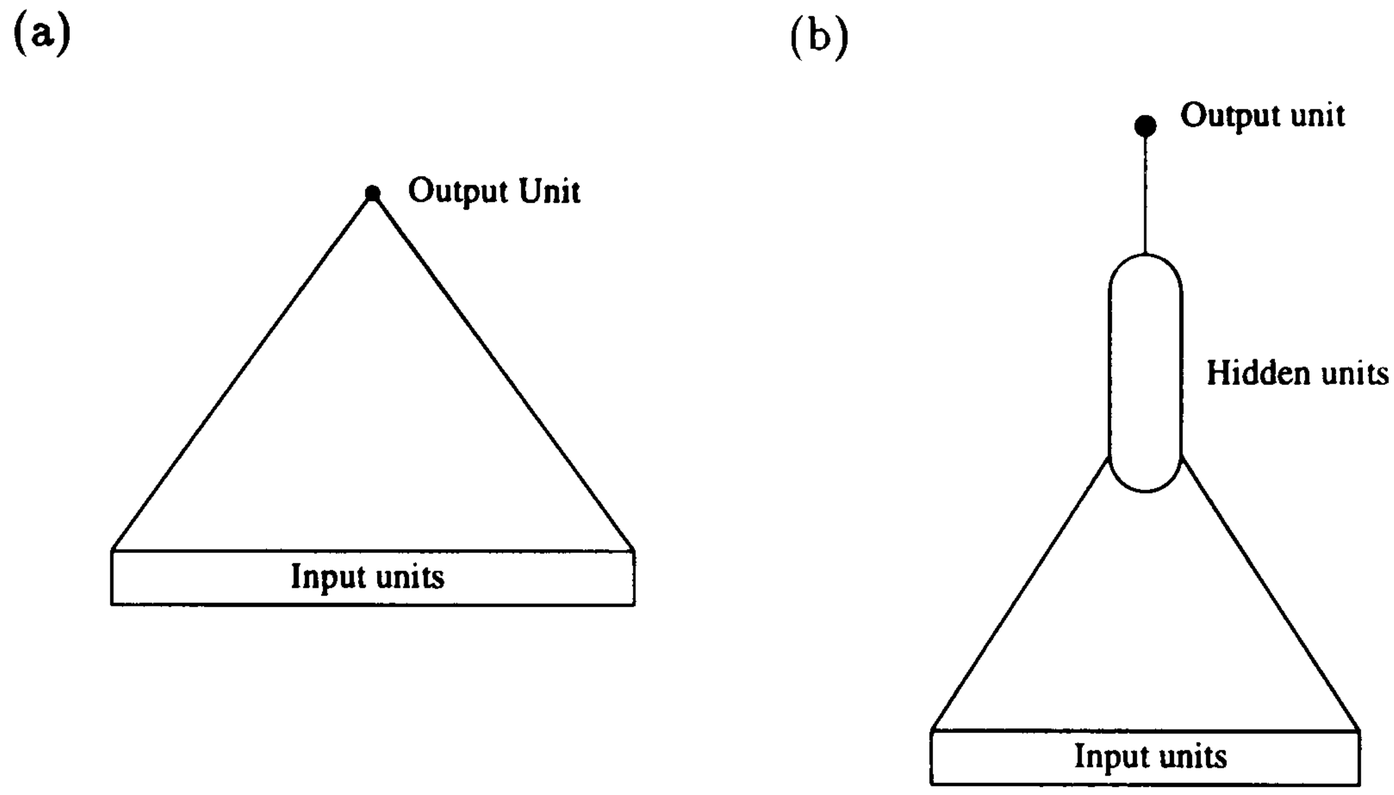


Figure 4.5: Filter networks: (a) Direct. (b) with hidden units.

network with a single output unit. As described in chapter 3, direct filter networks (fig. 4.5a) have a single layer of weights between the input layer and output units. Alternatively, a layer of hidden units may mediate between the input layer and output unit (fig. 4.5b).

The fan-in of the filter network is fixed at 31 receptors, corresponding to 93 degrees, for all animats in this and the next two chapters. In the intensity coding case, the filter network's input layer of weights is a 1-D array of size 31. In the multiscale coding case it is 2-D of size, 6×31, and in the rectified multiscale case it consists of 2, 2-D arrays of size 6×31.

In all coding cases, the convolution network consists of 15 identical copies of the filter network, centered at evenly spaced positions on the input array. This subsampling is only to save serial computation time, and would not be necessary if the animats were implemented on a parallel machine. The scalar output of the filter network in each position becomes an element in the motor array. Thus, the 1-D visual array is mapped into a 1-D, topography preserving, motor array. The motor array is used to stochastically determine the direction in which the animat moves a fixed distance of 10 on each time step as explained in chapter 3. The higher the value of a motor array element, the higher the probability that the animat will move in the corresponding direction. Figs. 4.6, 4.7 and 4.8 show the internal structure of animats in the three coding conditions.

4.4 Simulation method

The simulated circular arena had radius 128, and contained a circle of radius 10, surrounded by a goal region of radius 20 (fig. 4.1a). Because of the simplicity of the setup, nothing is gained by moving the circle between trials and so it remained in the center of the arena. In each trial, animats started at a random location. At each time step, the visual array was generated, and this was processed by the animat to produce the 15 element motor array. Based upon this, the

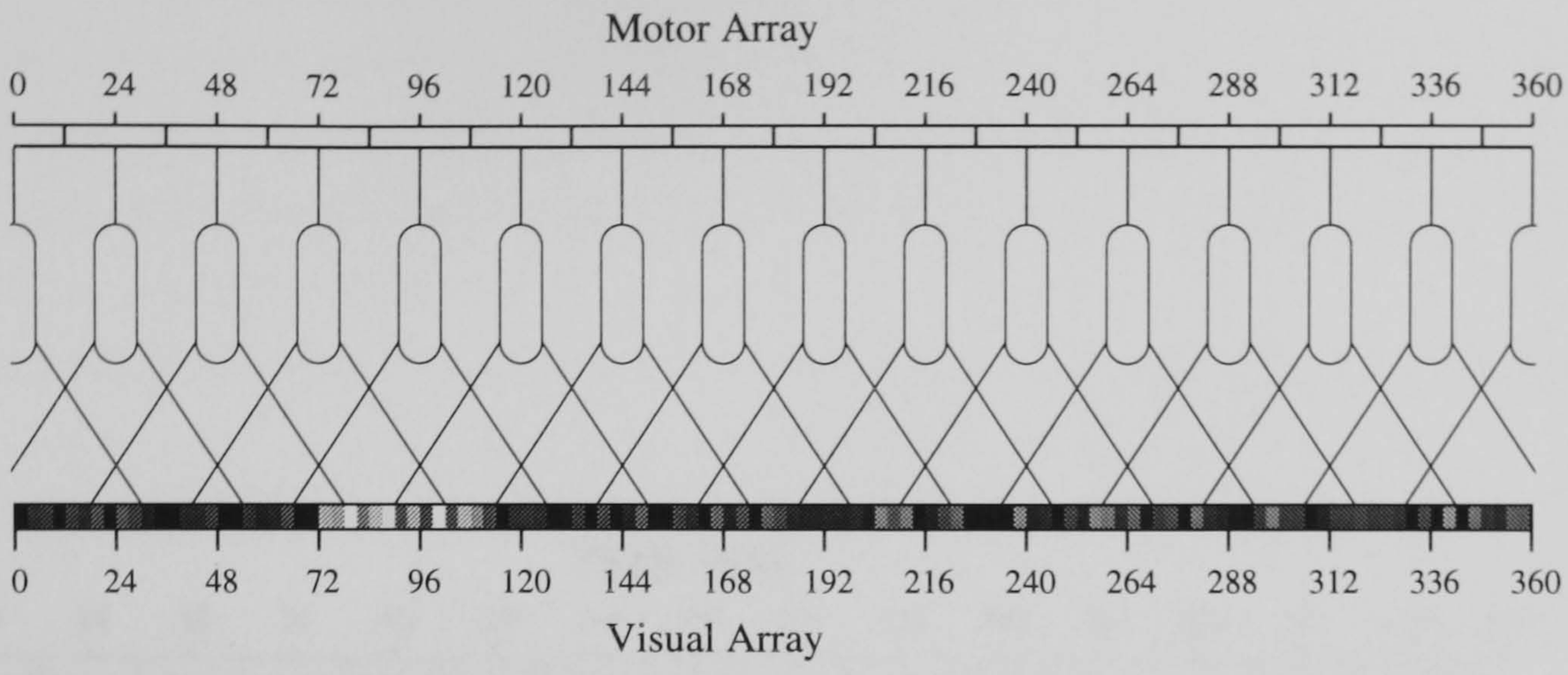


Figure 4.6: Animat sensorimotor system: intensity coding.
The 1-D visual array is convolved with the adaptive filter network to produce the 1-D motor array.

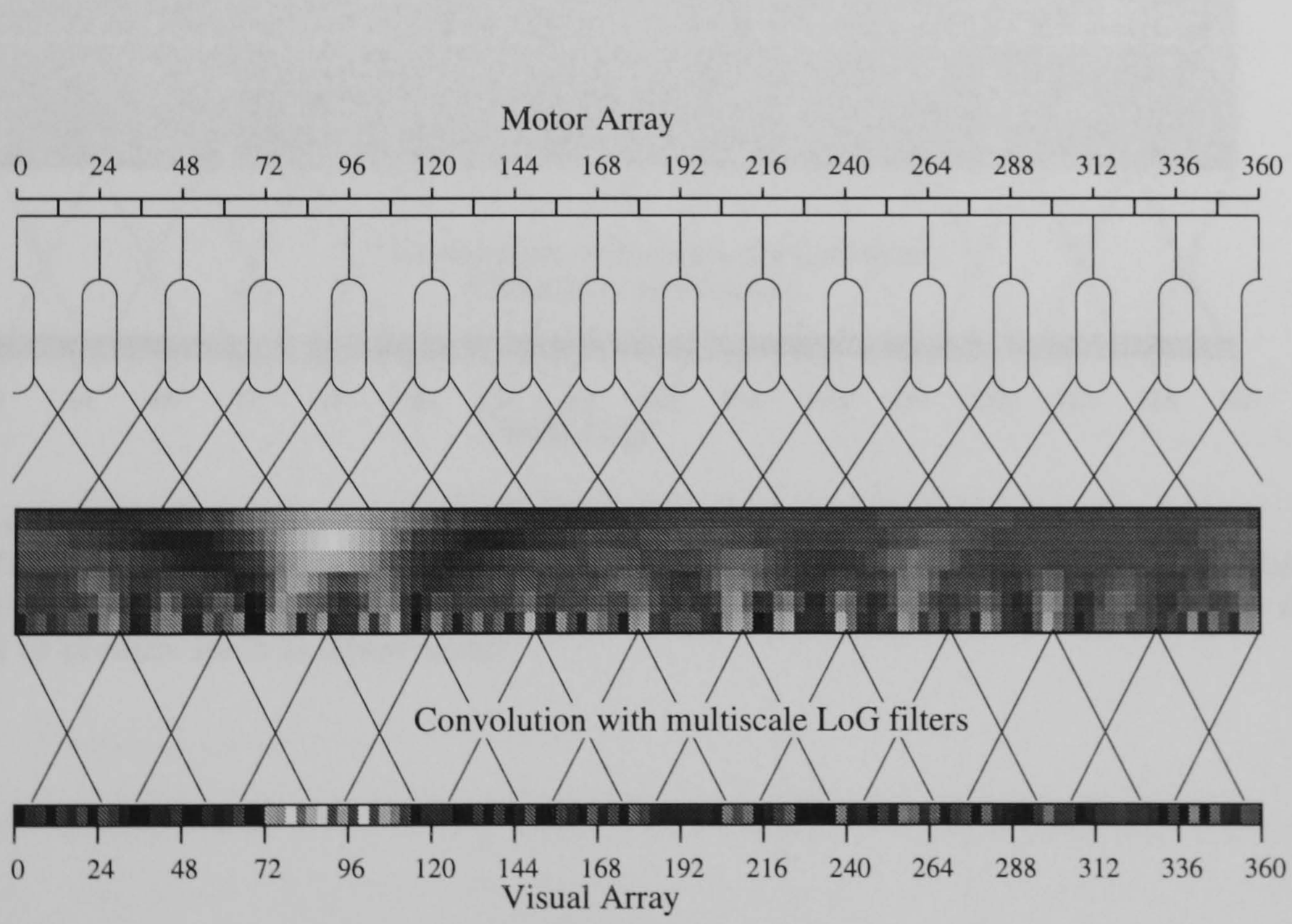


Figure 4.7: Animat sensorimotor system: multiscale coding.
The 1-D visual array is convolved by 6 LoG filters of different scale to produce a 2-D multiscale array. This is then convolved with the adaptive filter network to produce the 1-D motor array.

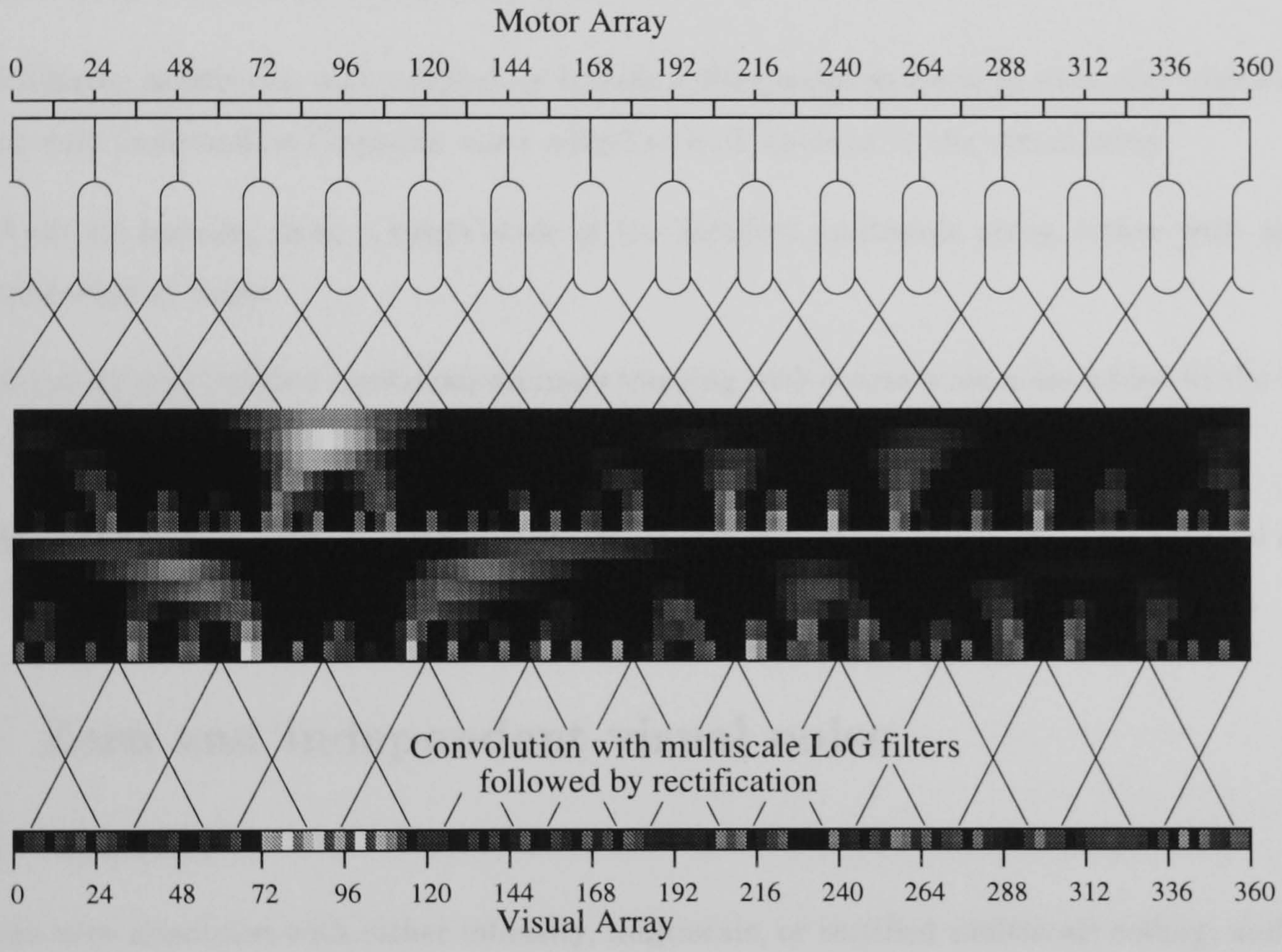


Figure 4.8: Animat sensorimotor system: rectified multiscale coding.
The 1-D visual array is convolved by 6 LoG filters of different scale to produce a 2-D multiscale array. Rectification produces two, 2-D arrays. These are then convolved with the adaptive filter network to produce the 1-D motor array.

direction of movement was stochastically determined as described in chapter 3, and animats moved a distance of 10 in that direction. A reinforcement signal of 0.0 was received on each time step, except when the animat entered the goal region, when a signal of 1.0 was received and a new trial begun. After each movement, filter network weights were modified according to the Q learning reinforcement algorithm as described in chapter 3. If animats had not reached the goal by 500 time steps, a new trial was begun.

In each coding and noise condition, animats with a range of filter network sizes were simulated. In each condition three animats were simulated with different random initial weights.

Three batches of simulations are reported:

- Intensity, multiscale, and rectified multiscale coding animats learning with zero visual noise, or with independent Gaussian noise added to each element in the visual array.
- Animats learning from a single scale of the rectified multiscale array, either with zero or independent noise.
- Intensity and rectified multiscale animats learning with coarse scale noise added to the visual array.

After learning, animats are examined in terms of performance, behaviour and internal structure.

4.5 Zero and independent visual noise

4.5.1 Method

Animats were simulated with either intensity, multiscale, or rectified multiscale coding, and filter networks with no hidden units (direct), or 2,4, or 8 hidden units. Animats learned with no visual noise, or with Independent Gaussian noise with mean 0.0 and standard deviation (sd) 0.05, or 0.10 added to each element in the visual array.

4.5.2 Performance

The performance (between 0 and 1) of an animat in a particular trial is measured as the minimum number of steps required to get from the start location to the goal, divided by the actual number of steps taken by the animat. The mean performance of an animat is obtained by testing over 1,000 (1k) trials in which the animat starts in random locations and wall and circle intensities are chosen randomly; the same conditions as during learning. Performance is between 0 and 1.0, with 1.0 corresponding to an animat taking the shortest route to the goal from any start location and

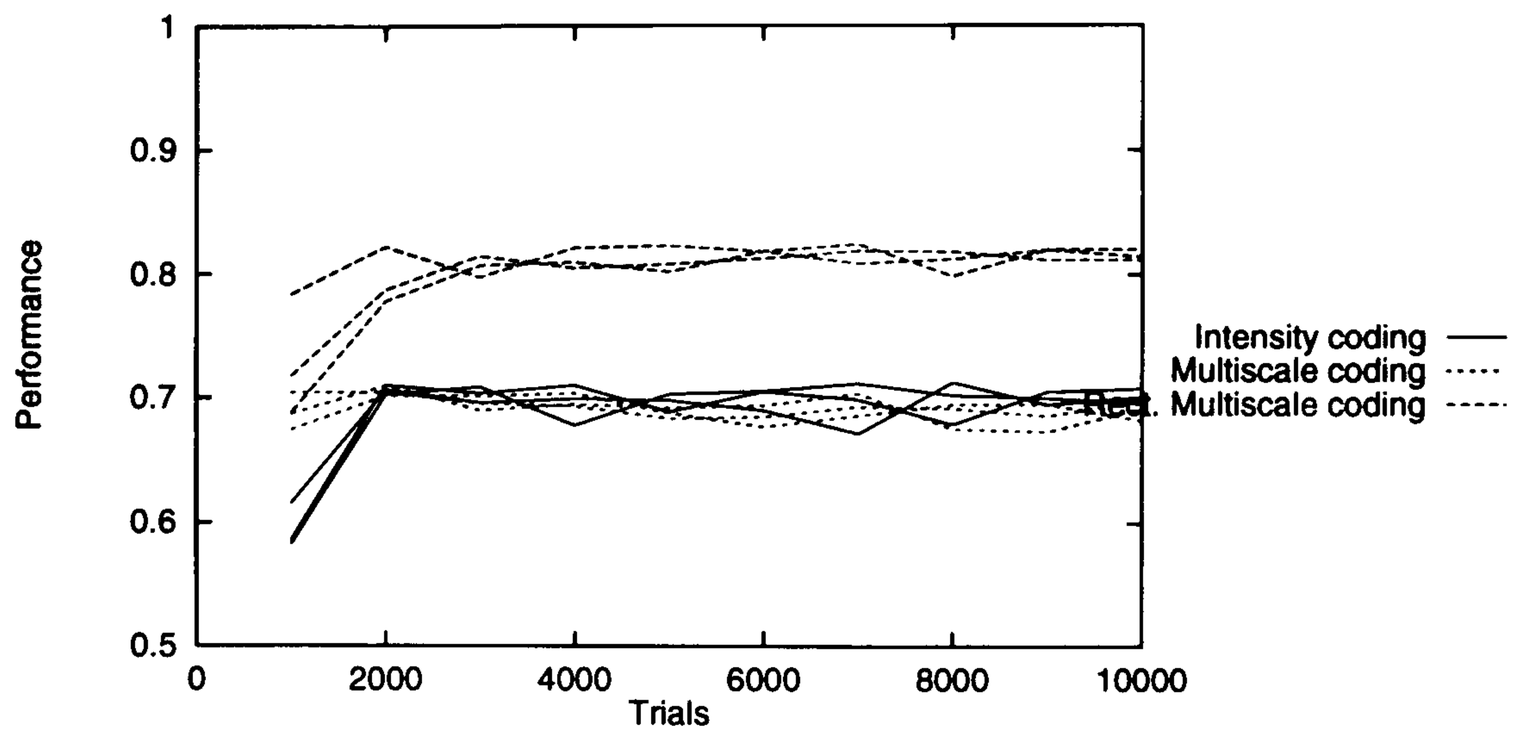


Figure 4.9: Learning curves for direct animats, learning with independent visual noise of sd 0.05. Three animats in each coding condition. Each data point is the mean performance of the animat over the preceding 1000 trials.

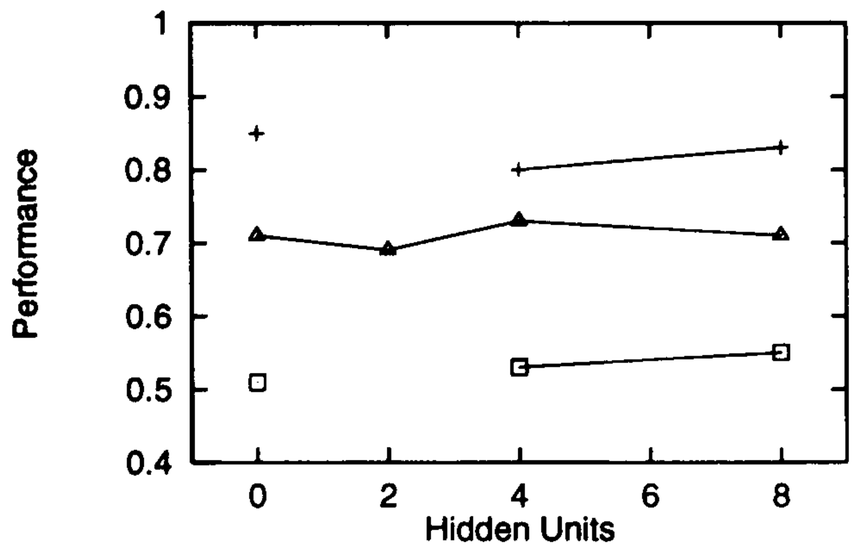
with any contrast. A performance of 0.5 means that the animat took, on average, twice as many steps to get to the goal as the shortest route.

Randomly behaving animats have a mean performance of 0.08±0.01. This figure was obtained by testing, over 1,000 (1k) trials with random start locations, an animat that randomly chooses one of the 15 directions in which to move on each time step. As with the learning animats, the maximum trial length was set at 500 steps.

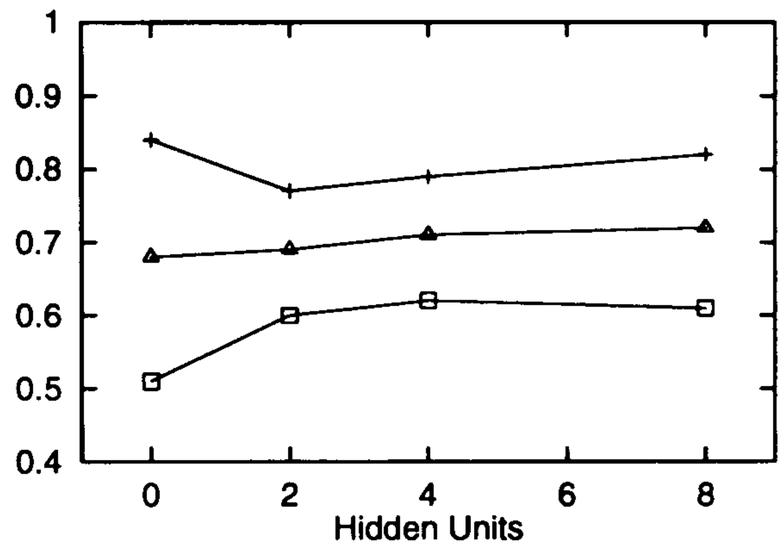
Fig. 4.10 shows the performance of animats after 10K learning trials, by which time animats in all conditions – except the two discussed below – had converged, and showed no signs of further improvement. Fig. 4.9 shows the learning curves of direct animats in the 0.05 noise condition. After learning, each of the three animats was tested over 1,000 (1k) trials without learning, and the mean performance over the three taken to provide the data point for each condition. Fig. 4.10a–c show performance as a function of noise and number of hidden units, for each of the three coding conditions (direct animats are plotted as zero hidden units). Note that a performance of 1.0 is not possible because the random choice of wall and circle intensities will on occasion lead to their difference being close to zero.

Direct animats with intensity coding perform at 0.85, 0.71 and 0.51 at noise sd 0.0, 0.05 and 0.1 respectively. With two hidden units, when noise sd is 0.05, performance is at the same level as for direct animats. However, at zero and 0.1 noise levels, learning is unstable with performance wildly oscillating over time. With 4 or more hidden units, performance is stable and at the same level as for direct animats. It is the only case of unstable learning in these simulations. This is a puzzling result and therefore deserves further attention. Replication of animats in these conditions led to the same result, and performance did not settle down when learning was extended to 50k trials. Given that direct animats perform as well as any other intensity coding animats, the 2 hidden unit results are not crucial to the comparison of performance across coding scheme that is

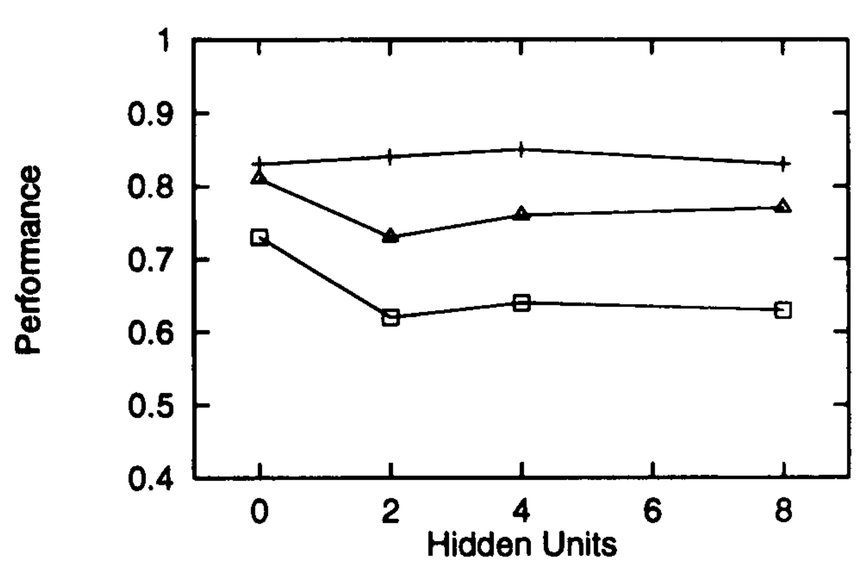
(a) Intensity coding.

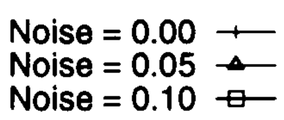


(b) Multiscale coding.

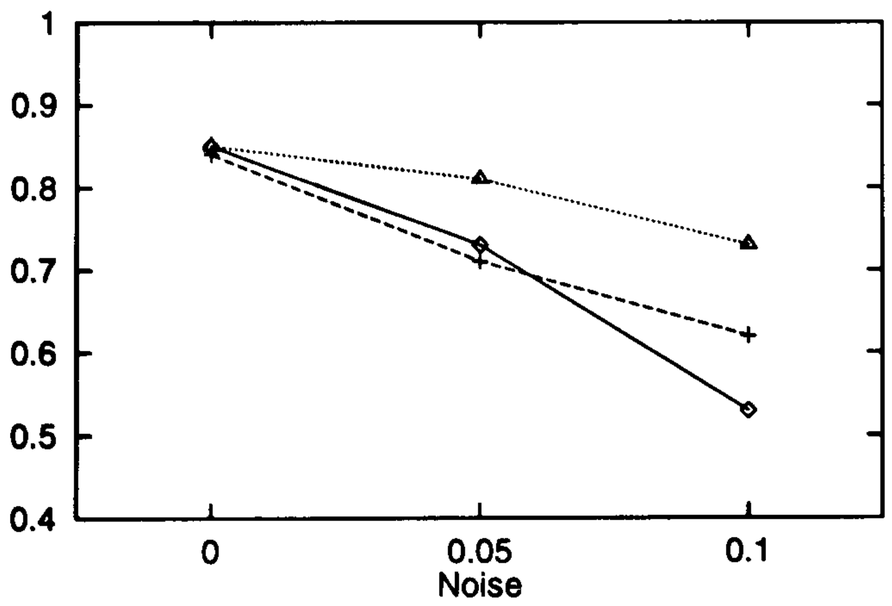


(c) Rectified Multiscale coding.





(d) Comparison across coding.



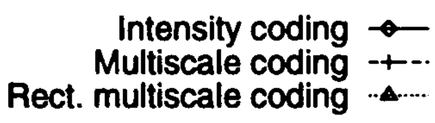


Figure 4.10: Mean performance of animats on the circle approaching task. (a)–(c) plot performance as a function of noise and filter network size in each of the three coding conditions. Direct networks are plotted as 0 hidden units. (d) Plots the performance of the highest performing animats in (a)–(c) as a function of noise. Each data point is the mean of three animats. Animats tested at the same level as during learning. All standard error$\leq$0.04 for intensity coding, and $\leq$0.03 for multiscale and rectified multiscale coding.

the main focus of this work. Therefore, these isolated instabilities are not examined further.

Direct multiscale coding animats perform as well as those with hidden units at noise levels of 0.0, where they achieve 0.84 performance, and at noise 0.05, where they achieve 0.68 performance. With noise at 0.1, 4 hidden units are required to achieve a performance of 0.62, which is not exceeded with 8 hidden units.

With rectified multiscale coding, direct animats perform at least as well as those with hidden units at all noise levels, achieving performances of 0.83, 0.81 and 0.73 at noise levels 0.0, 0.05 and 0.1 respectively.

Fig 4.10d compares performance of animats with the different coding conditions as a function of noise level. Here, in each noise and coding condition the highest performing animats are plotted, regardless of the number of hidden units. This is justified because the different coding schemes may impose different computational demands and so require different sizes of network for optimal performance. Except for multiscale coding animats at noise level 0.1, the highest performing animats in each condition are direct.

At zero noise, there is no difference in performance between any of the coding schemes. At noise = 0.05, rectified multiscale animats perform significantly better than intensity or multiscale animats (t test: $t = 16.959$, $p<0.001$), between which there is no difference. When noise = 0.1, rectified multiscale again significantly outperforms the others ($t=10.855$, $p<0.001$), between which multiscale coding significantly outperforms intensity ($t=7.750$, $p<0.01$).

These results indicate that regardless of coding, animats are able to learn to perform at a level far above chance. Rectified multiscale coding leads to significantly better performance when independent visual noise is present than either intensity or multiscale coding. At the highest noise level, intensity coding is significantly worse than both multiscale and rectified multiscale coding.

Contrast

Fig. 4.11 plots performance as a function of the magnitude of contrast for direct animats learning (and tested) with zero noise and independent noise of sd 0.1. Without noise, animats perform well at all contrast levels except the lowest and there is no difference between coding. However, with noise, rectified multiscale coding animats are far less degraded as contrast decreases than intensity or multiscale coding animats. Thus, rectified multiscale coding facilitates performance in situations of noise and low contrast.

Having examined the performance differences across the various conditions, the learned computations underlying these performances are now examined. There are two aspects to this: the behaviour of the animats in the environment, and the learned weights and biases of the network underlying their behaviour.

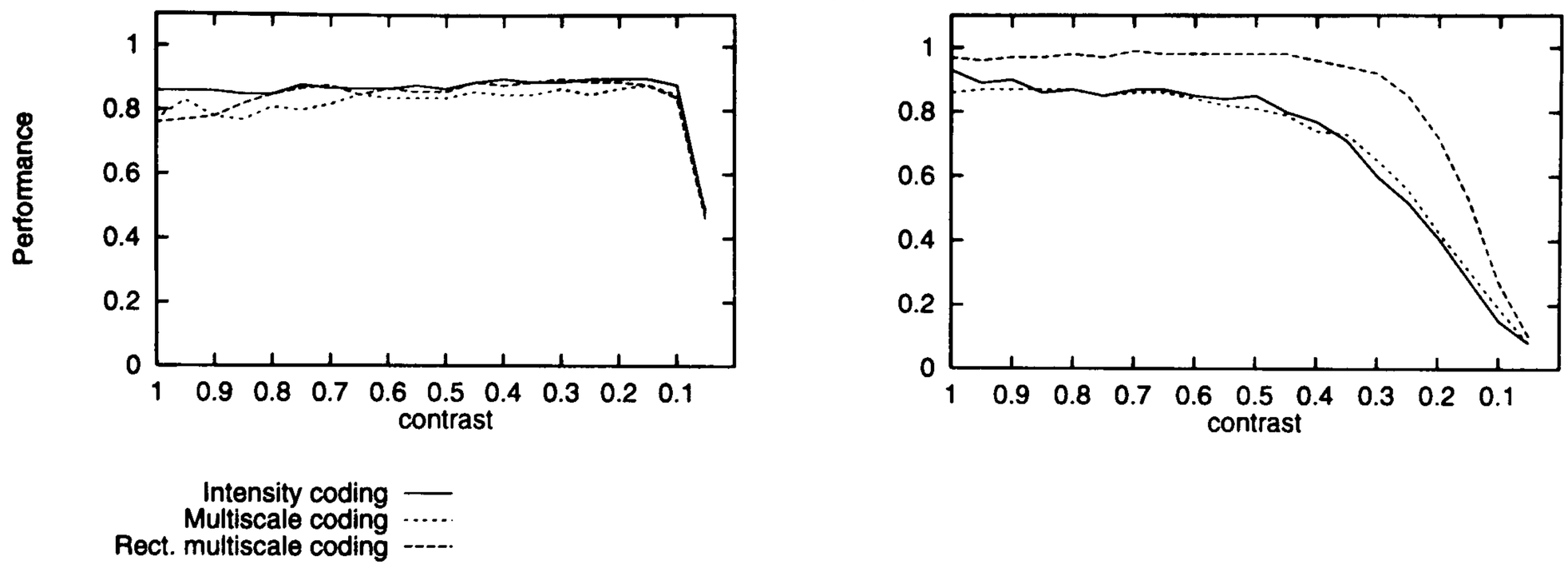


Figure 4.11: Performance as a function of contrast for direct animats. Animats tested at the same noise level as during learning (a) zero noise, (b) Independent noise of sd 0.1
Over 3000 test trials, animat performance was split into twenty groups according to the magnitude of the difference between circle and wall intensities (in 0.05 steps).

4.5.3 Behaviour

Fig 4.12a shows typical paths of an animat before learning. Animats behave like the random animat before learning, although the initial random weights tend to instill a slight behavioural bias. Because the initial weights are small, the stochastic aspect of movement direction choice outweighs that due to the network.

Fig 4.12b and c show typical paths for direct intensity coding animats after learning without noise and with independent visual noise of sd 0.10. Animats are tested under the same noise conditions as during learning, and the contrast is at the low level of 0.2. In the noiseless case, animats move in a curve toward the goal region and approach the edge of the circle. Animats that learned with noise show similar behaviour although the paths are much less efficient; hence the 0.34 difference in performance. Similar behavioural strategies of curving toward the edge of the circle are shown by all the intensity coding animats.

Fig 4.12d and e show typical paths for direct rectified multiscale animats. In the noiseless condition, animats behave like the intensity coding animats, though their paths tend to be somewhat less curved. With noise, rectified multiscale animats move toward the center, rather than the edge of the circle. Clearly rectified multiscale animats are less affected by the noise than intensity coding animats. As may be expected, movement is less efficient with increasing distance from the circle.

The behaviour of animats tested in a larger arena and with two circles of the same contrast is shown in fig. 4.13, which clearly shows the different behavioural strategies of intensity and rectified multiscale coding animats. The intensity coding animat shows a stereotyped path curving toward the circles edge whereas the rectified multiscale coding animat more directly moves toward the center of the circle. As expected, increasing the radius of the circle makes it more attractive. In

environment containing circles of fixed radius and contrast, animats typically move toward the nearest, since this will elicit the higher output. More generally, in multi-circle environments, the relative distance, radius, and contrast of the circles will determine which one the animat tends to steer toward.

4.5.4 Internal Structure

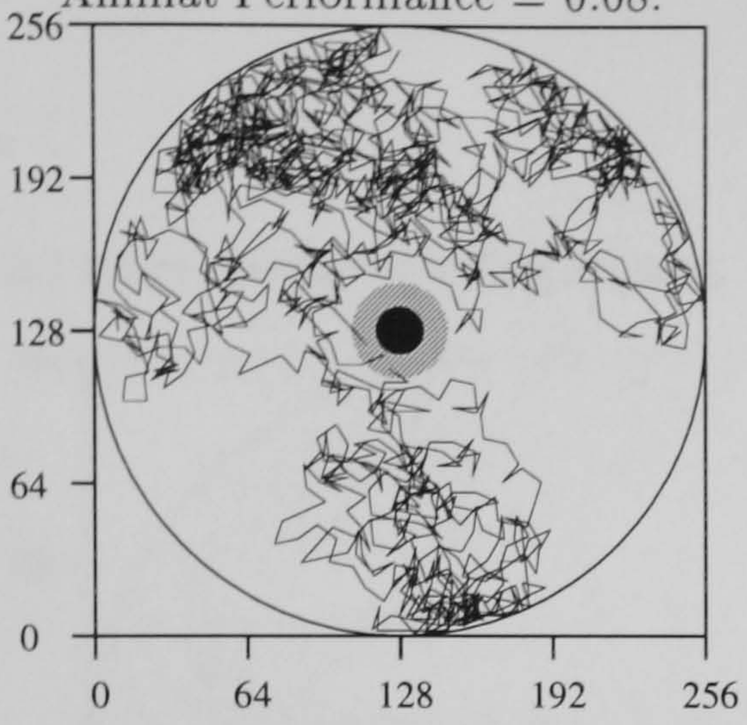
Intensity Coding

Turning now to the learned internal structures underlying these behaviours, the left hand column of fig. 4.14 shows the weights of filter networks controlling direct intensity coding animats at the three noise levels. The output unit sums over these, adds the bias, and puts the result through a sigmoid function to determine output activation. Figs. 4.14a and c are for the animats whose behaviour was shown and discussed above. The weights are positive on one side of the axis and negative in the other, summing to around zero, and the output units have a small negative bias. As the noise levels increases, the magnitude of the weights decreases, whilst keeping the same structure; this same weight structure was learned by all direct intensity coding animats. Because the weights are roughly balanced, the output unit does not respond when the visual input has the same intensity across the array, regardless of its absolute value. The negative bias ensures that slight deviations from a flat input, due to noise, still result in a null response. However, when there is a difference in intensity between the two sides, the network produces a non null response.

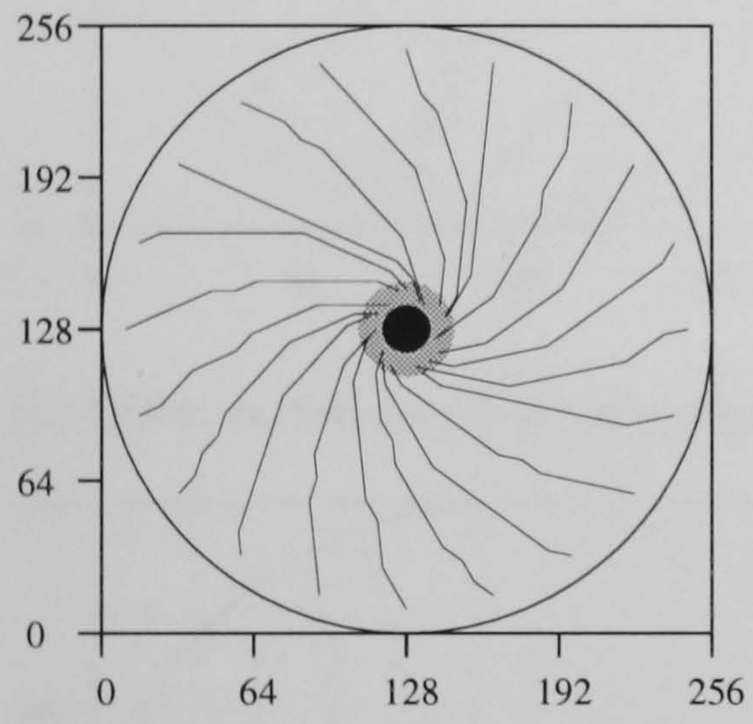
The right hand column of fig. 4.14 plots the output of the filter networks in response to the circle centered at the distance and bearing given by the axes. The bearing of the circle is relative to the center of the filter networks receptive field, and so at zero bearing the circle is directly facing the filter network. White corresponds to output of 1.0, black to output of 0.0; gray levels are comparable across plots. When the circle directly faces the filter network, the balanced weight structure results in a null output. When the circle is off-center, the summed intensity will be greater on one side of the filter axis than the other and as a result, the output unit will have a non-null response. When located at the other side of the filter axis, the circle at the same contrast produces an output response in the opposite direction. As can be seen in figs. 4.14b and c, the smaller magnitude weights produced by learning with noise result in a weaker form of the same response pattern.

These filters have learned a differential like shape that is insensitive to absolute intensity level, but is most sensitive to the spatial change in intensity around the edge of the circle's image. Such step shaped filters respond most strongly when the image of the circle covers one half of their receptive field, and produce a null response when the circle is in the center of the receptive field, hence the tendency for these animats to steer the curved path toward the circle edges shown in

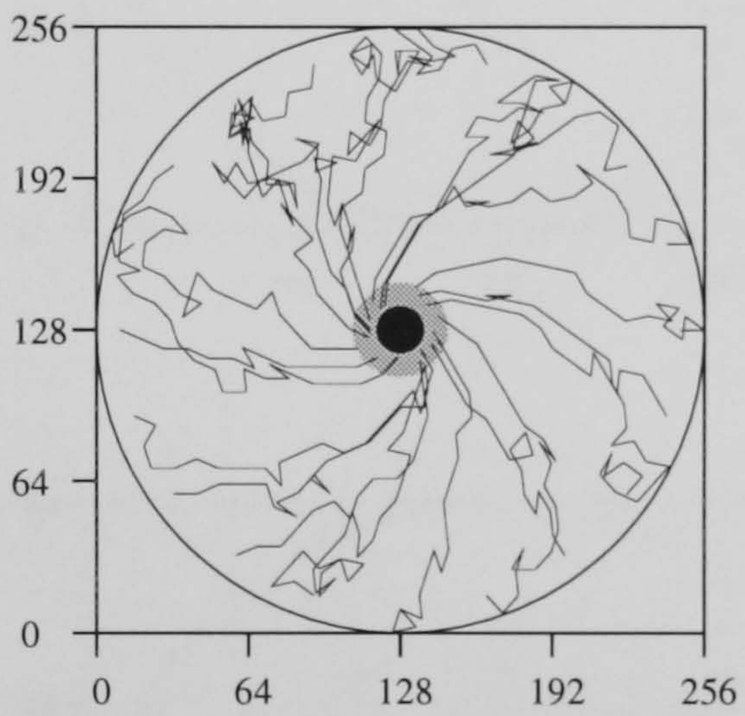
(a) Before Learning
Animat Performance = 0.08.



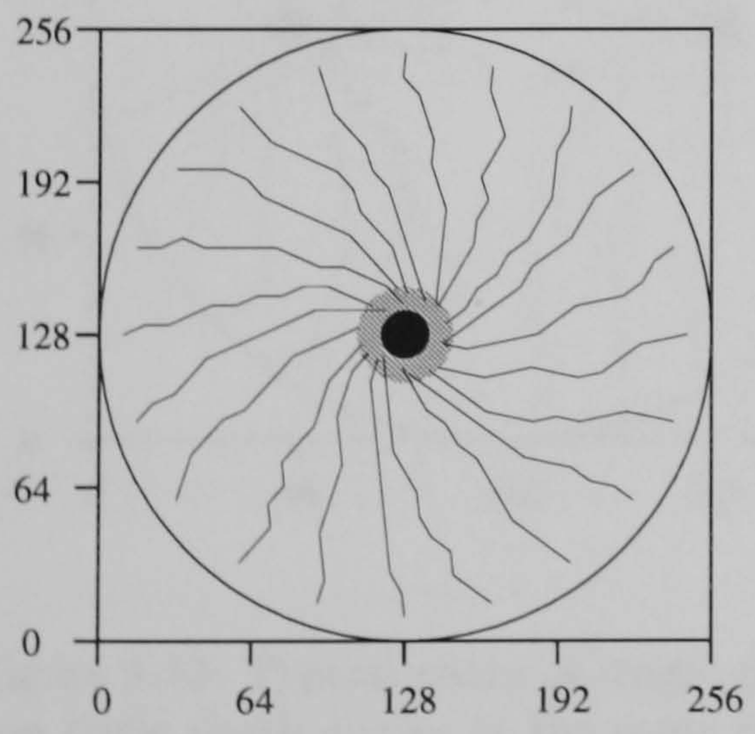
(b) Intensity coding. Noise = 0.0
Animat Performance = 0.85



(c) Intensity coding. Noise = 0.1
Animat Performance = 0.51



(d) Rectified multiscale coding. Noise = 0.0
Animat Performance = 0.82



(e) Rectified multiscale coding. Noise = 0.1
Animat Performance = 0.73

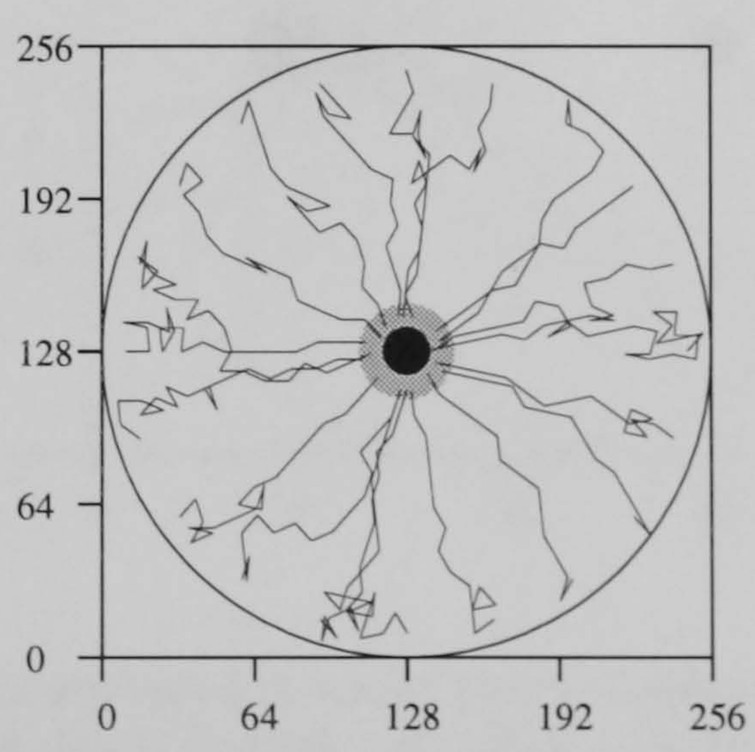
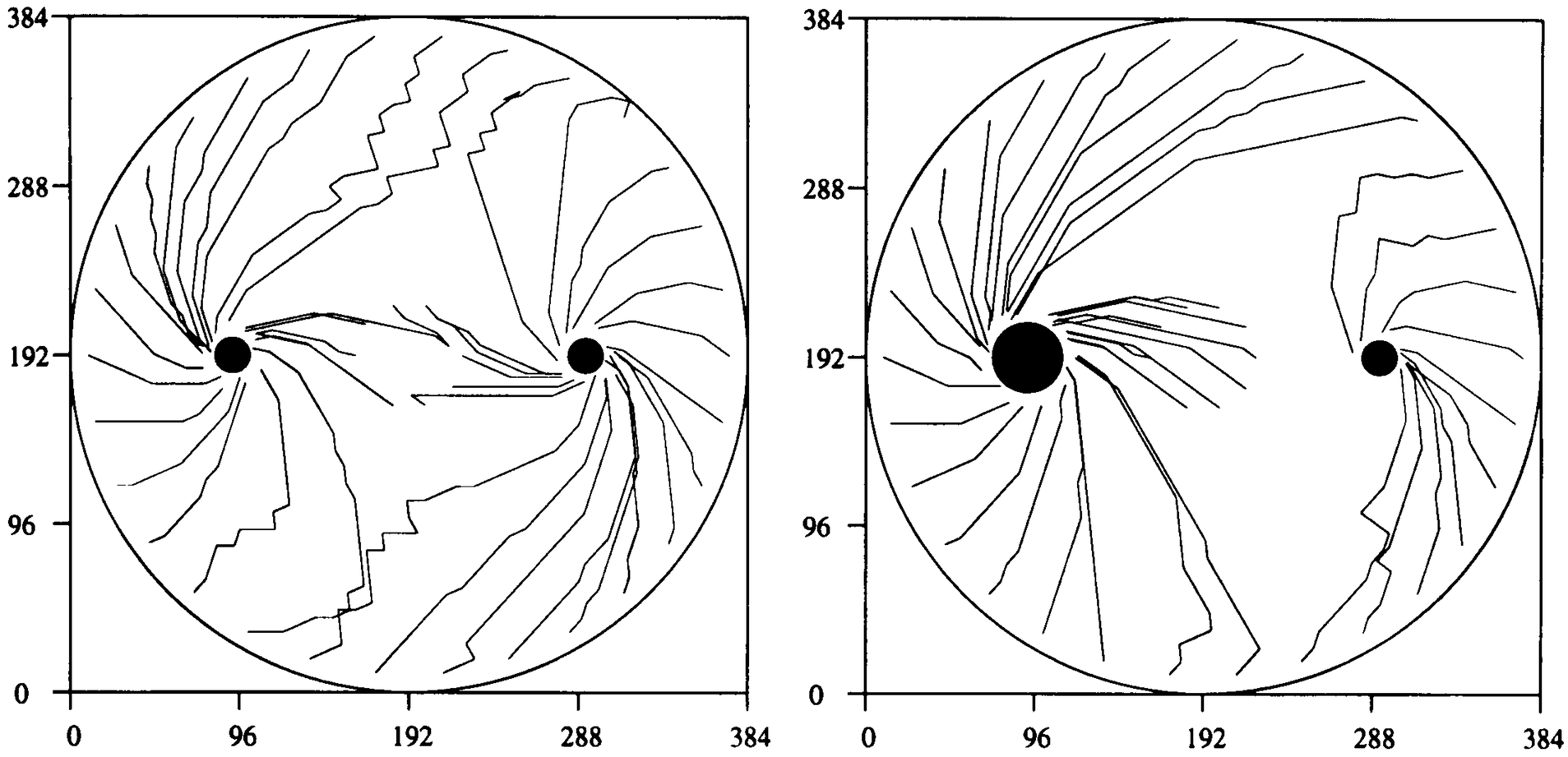


Figure 4.12: Typical individual animat paths, tested at the same noise level as during learning. The invisible goal region is shown in gray. Performance is of the individual animat rather than the mean performance for animats in the condition.
(a) Before learning (ie approx. random animat).
(b) and (c) after learning: Direct intensity coded animat at independent noise levels 0.0 and 0.1.
(d) and (e) after learning: Direct rectified multiscale coding animat at independent noise levels 0.0 and 0.1.
(For these plots: circle intensity = 0.4, wall intensity = 0.2. (a) 4 paths. (b)–(e) 20 paths).

(a) Direct intensity coding animat.



(b) Direct rectified multiscale coding animat

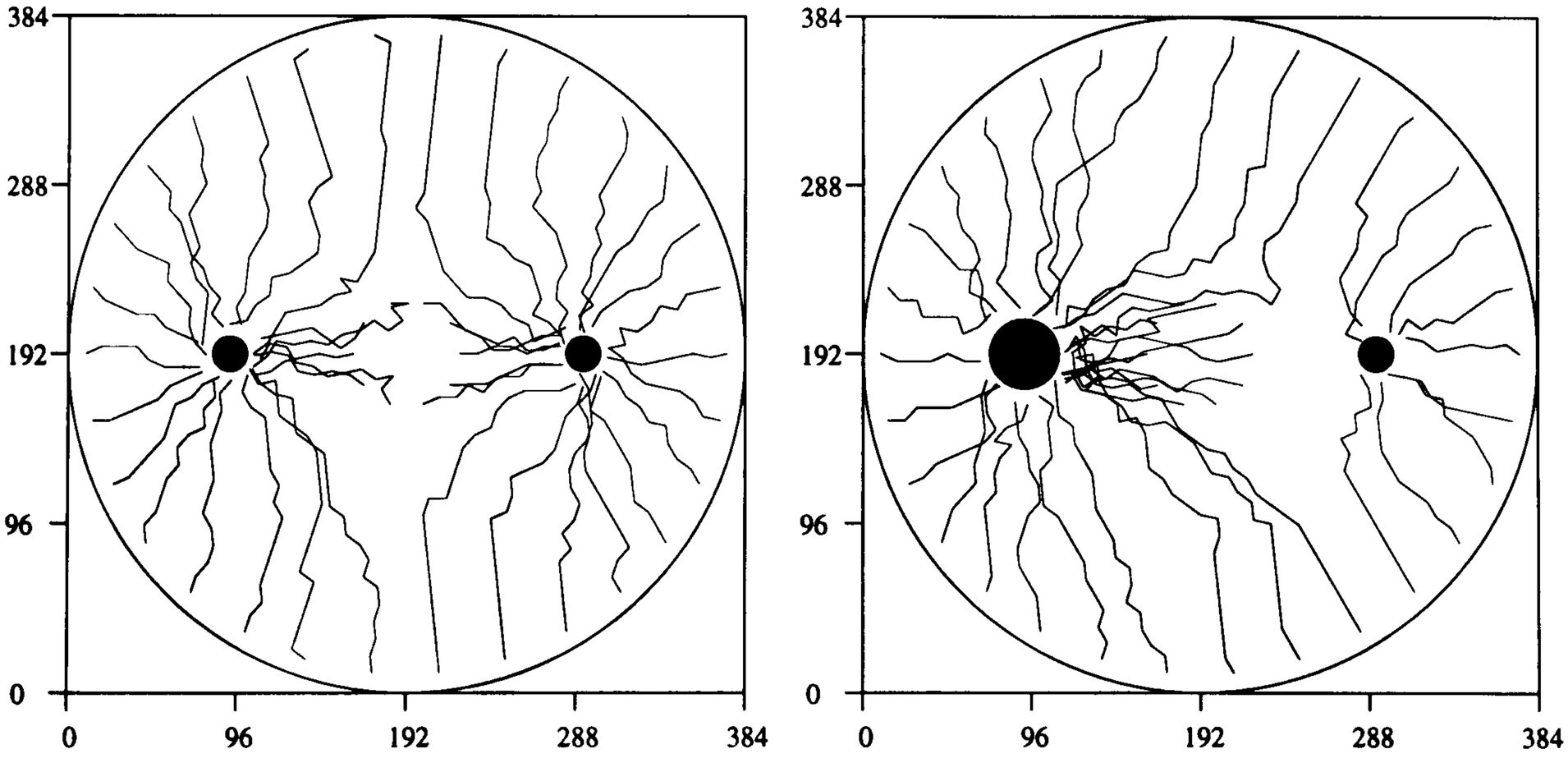


Figure 4.13: Typical paths of single animats animats tested in larger arena containing more than one circle (both circles at the same contrast). Animats learned and tested at independent noise level 0.05. Circle intensity = 0.7, Wall intensity = 0.2.

fig. 4.12b and c.

Greater understanding of learned animats can be obtained by comparing their performance with that of hand-wired animats. The latter will provide a set of performance scores that can provide a comparative context for the scores of learned animats. In addition, animats can be hand-wired so that they capture the important aspects of the computation learned by the reinforcement animats. In test, these should perform at least as well as their learned counterparts. If this is not the case, then it indicates that the important computations have not been captured by the hand-wired animat. Thus, the hand-wired animats provide a useful empirical check on explanations of what animats have learned to compute.

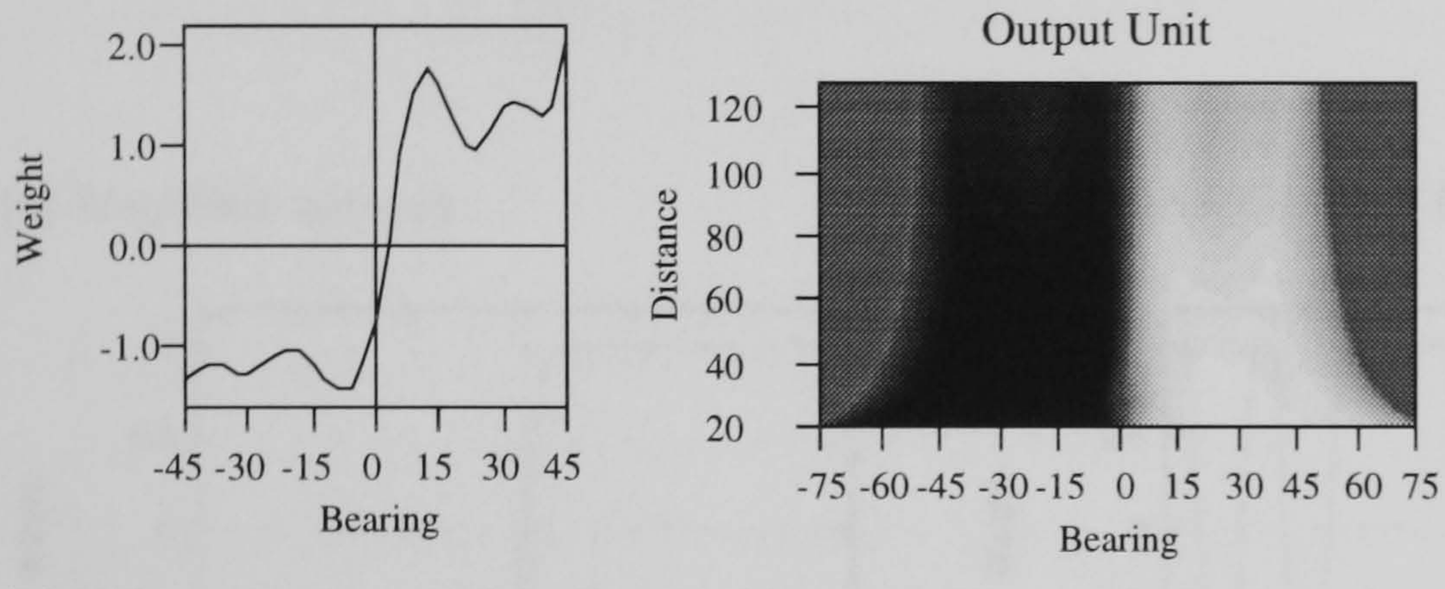
Fig. 4.15a shows the weights of two direct, hand-wired animats. On the left, the filter is balanced, with positive weights on one side of the axis, and negative weights on the other. This is the weight structure learned by direct, intensity coded animats. The network on the right is also balanced, but has no coherence in the arrangement of weights, with each one randomly either positive or negative. Thus, like the learned animats, they are insensitive to absolute intensity, but are not particularly structured to respond to the coherent patches of contrast corresponding to the image of the circle in the present task.

Fig. 4.15a plots the performance of the hand-wired animats as a function of noise, together with the performance of direct intensity coded animats after learning (as fig.4.10). With no noise, and at noise = 0.05, step nets and the learned nets perform at about the same level, reflecting their similar structure. At noise = 0.1 however, the learned animats perform more than 0.1 lower than step networks. At all noise levels, the random balanced network performs substantially worse than the others. As noise increases, the difference between this net's performance, and the step animats increases. The difference between the learned animats performance and that of the step net at noise = 0.10 presumably arises because the learned weights have such low magnitude (see fig. 4.14). Thus, reinforcement learning has led to a less than optimal animat in the high noise condition. However, the performance of the step net at both noise conditions is substantially lower than that of direct rectified multiscale animats.

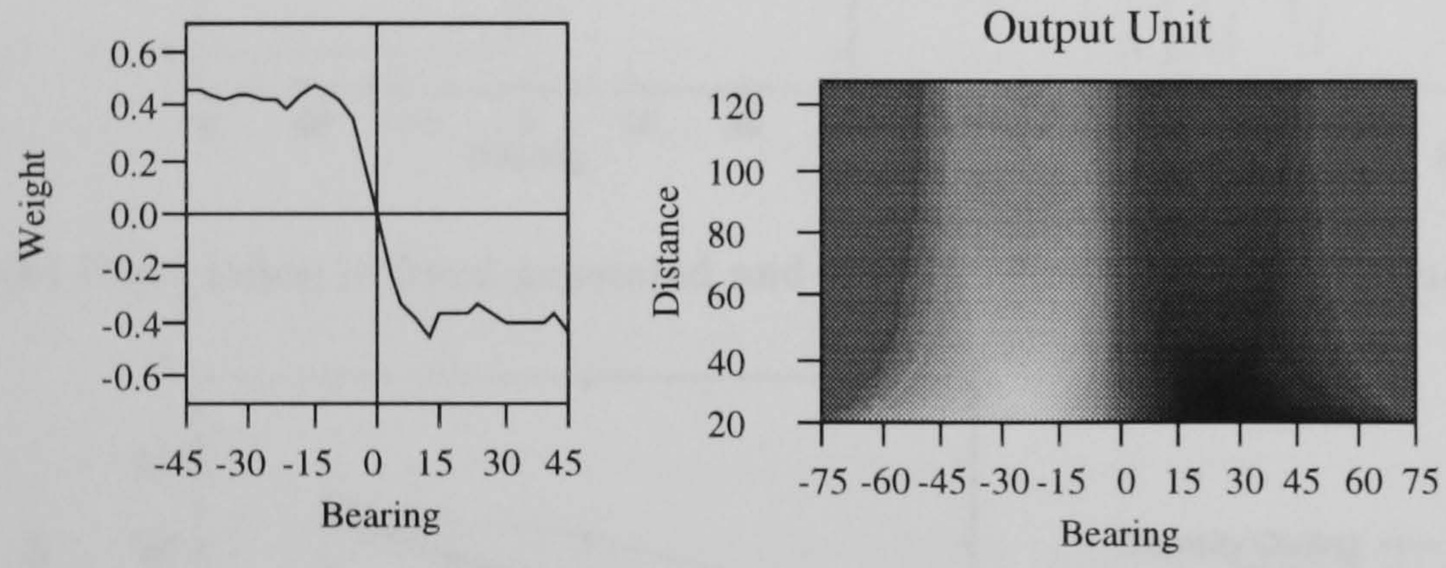
Rectified multiscale coding

Fig. 4.16 shows the learned filter network weights and corresponding output activation patterns of direct rectified multiscale coding animats. Figs. 4.16a and c are for the animats whose behaviour is shown in figs. 4.12d and e and discussed above. The left column plots the filter network weights: a 2-D array of weights for both the positive and negative components of the multiscale filtered visual array. The network output unit sums over these, the bias is added and the result put through a sigmoid function to determine output activation. The right column displays output activation in response to the circle. White corresponds to output of 1.0, black to output of 0.0; gray levels are

(a) Noise = 0.0. Output unit bias = -0.15. Animat performance = 0.84.



(b) Noise = 0.05. Output unit bias = -0.21. Animat performance = 0.71.



(c) Noise = 0.10. Output unit bias = -0.52. Animat performance = 0.51.

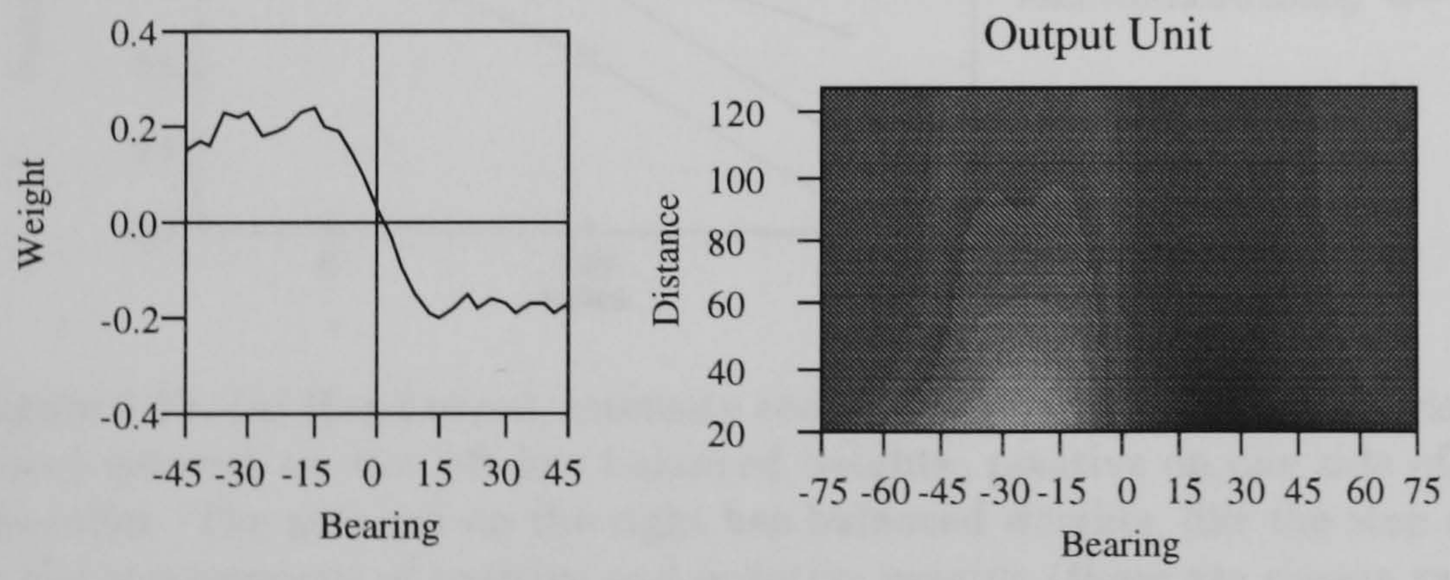
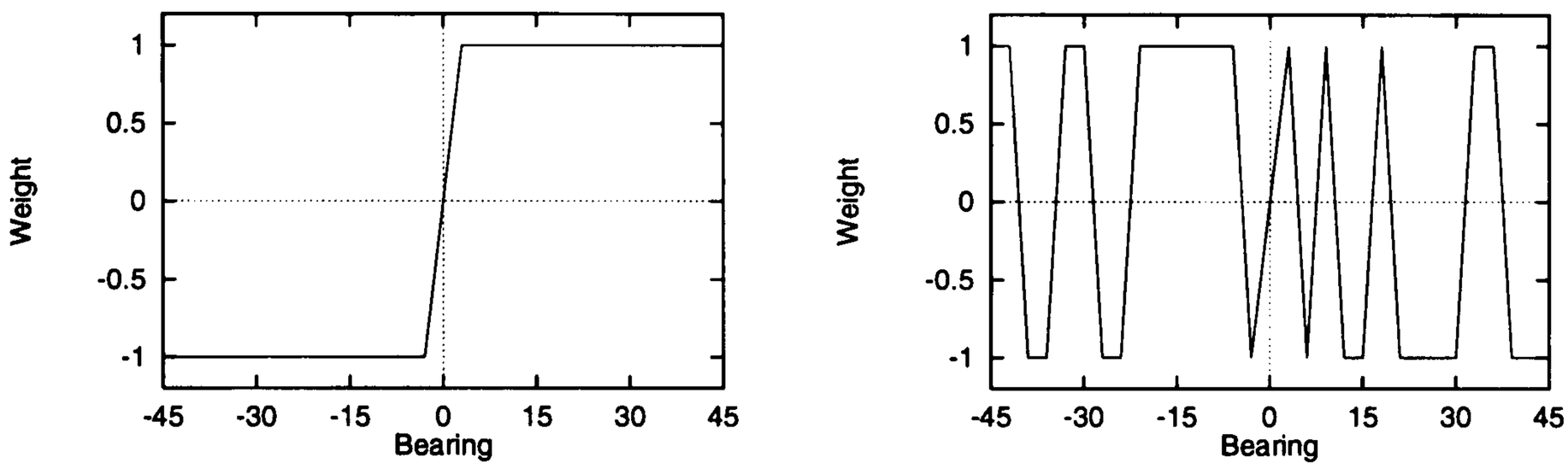


Figure 4.14: The learned weight structure and response patterns of direct intensity coding filter networks. (a) Noise = 0.0 (b) Noise = 0.05 (c) Noise = 0.10.
The left-hand columns show the learned weights and output unit bias.
The right-hand column shows the activation of the output unit in response to the circle as a function of its distance and bearing relative to the center of the filter network's receptive field.
(For these plots: circle intensity = 0.7, wall intensity = 0.2, noise = 0.0)

(a) Step filter network.

(b) Random balanced filter network.



(b) Comparison of hand generated and learned, direct intensity coding animats.

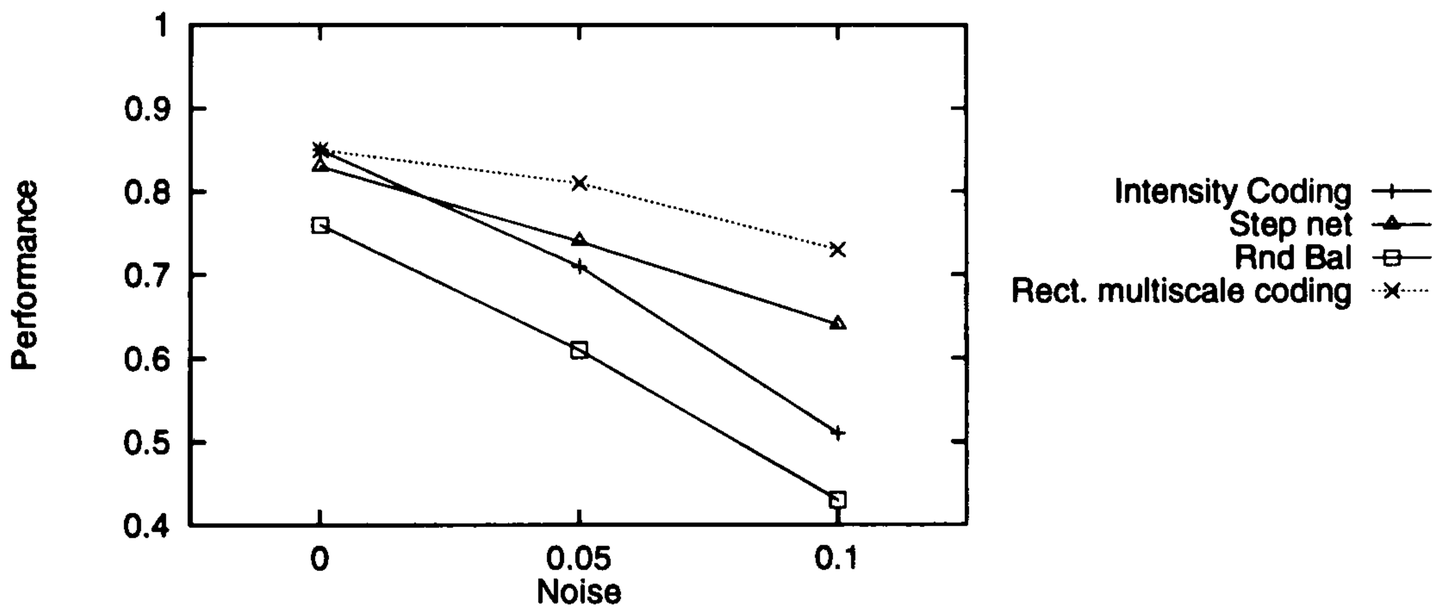


Figure 4.15: (a) Hand wired, intensity coding filter networks for comparison with learned ones. The direct network on the left has balanced weights: positive on one side of the axis, and negative on the other. The network on the right has balanced weights, like the step network, but no coherence in the arrangement of positive and negative weights (these are chosen randomly). (b) Comparison of performance of animats made up of the above networks with learned, direct intensity coding animats. Direct rectified multiscale coding performance also shown.

comparable across plots.

With zero sensory noise, large weights are concentrated at the coarsest scales in both the positive and negative arrays (figs. 4.16a). With noise at 0.05, large weights are concentrated at the coarsest 2 scales, and around the central region. At the highest noise level, large weights are almost exclusively at the coarsest scale, again around the central region. Very similar weight structures were learned by all direct rectified mulitscale animats learning with independent noise.

As was shown in fig. 4.3, the coarse scales are maximally active when the circle is nearby and hence subtends a large angle; when the circle is far away, and hence subtends a small angle, it elicits coarse scale activity of a smaller amplitude, but over a wider region. Thus within the range of subtended angles and filter scales used here, the circle elicits coarse scale activity regardless of distance. The independent noise, in contrast, causes most activity at the finest scale, and less activity with increasing scale. The rectified multiscale filter networks have learnt to exploit this by using the coarsest scale to detect the circle, since this is the scale least affected by noise. Hence the much smaller decrease in performance of rectified multiscale coding animats as the noise level increases compared with intensity coding animats.

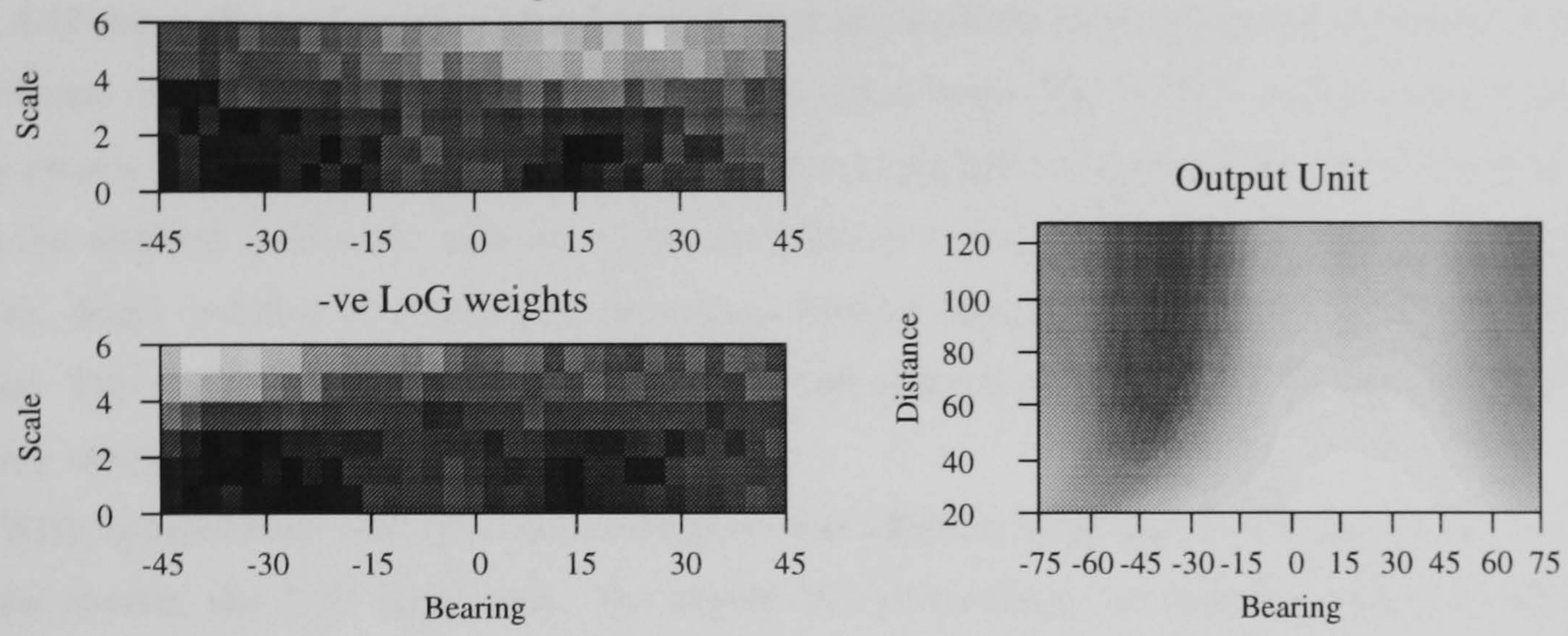
4.6 Rectified single scale coding

Examination of the learned internal structure of rectified multiscale coding animats has suggested that their robustness when learning in the presence of independent noise occurs because they learn to detect the circle using the output of coarse scale LoG filters, which are relatively unaffected by independent noise, whilst still responding to the circle. If this is the case, then animats having only the coarse scales as input should learn to perform as well as those with multiple scales, and better than animats having only fine scale input. The following simulations test this hypothesis.

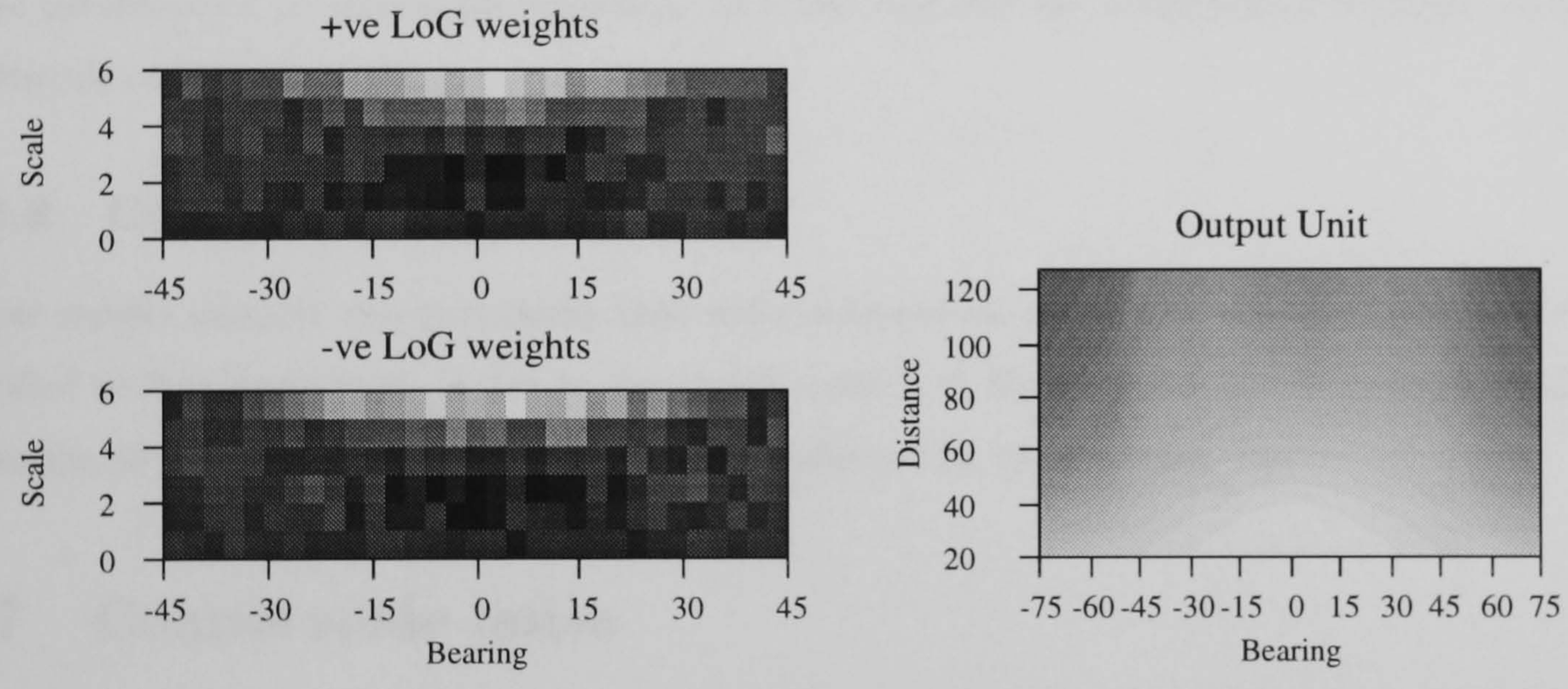
4.6.1 Method

Direct animats were simulated with rectified single scale LoG filters, each one corresponding to one of the scales in the rectified multiscale case. The sensory coded array of each single scale animat is identical to one of the rows of the 2-D, multiscale array. Rectification of the single scale filtered array yields two, 1-D arrays. Thus, rectified single scale coding animats have a subset of the input of rectified multiscale animats. Three animats were simulated in each scale and noise condition. Learning continued for 10k trials, by which time all had converged.

(a) Noise = 0.0. Output unit bias = -0.09. Animat performance = 0.82.



(b) Noise = 0.05. Output unit bias = -0.45. Animat performance = 0.81.



(c) Noise = 0.10. Output unit bias = -0.51. Animat performance = 0.74.

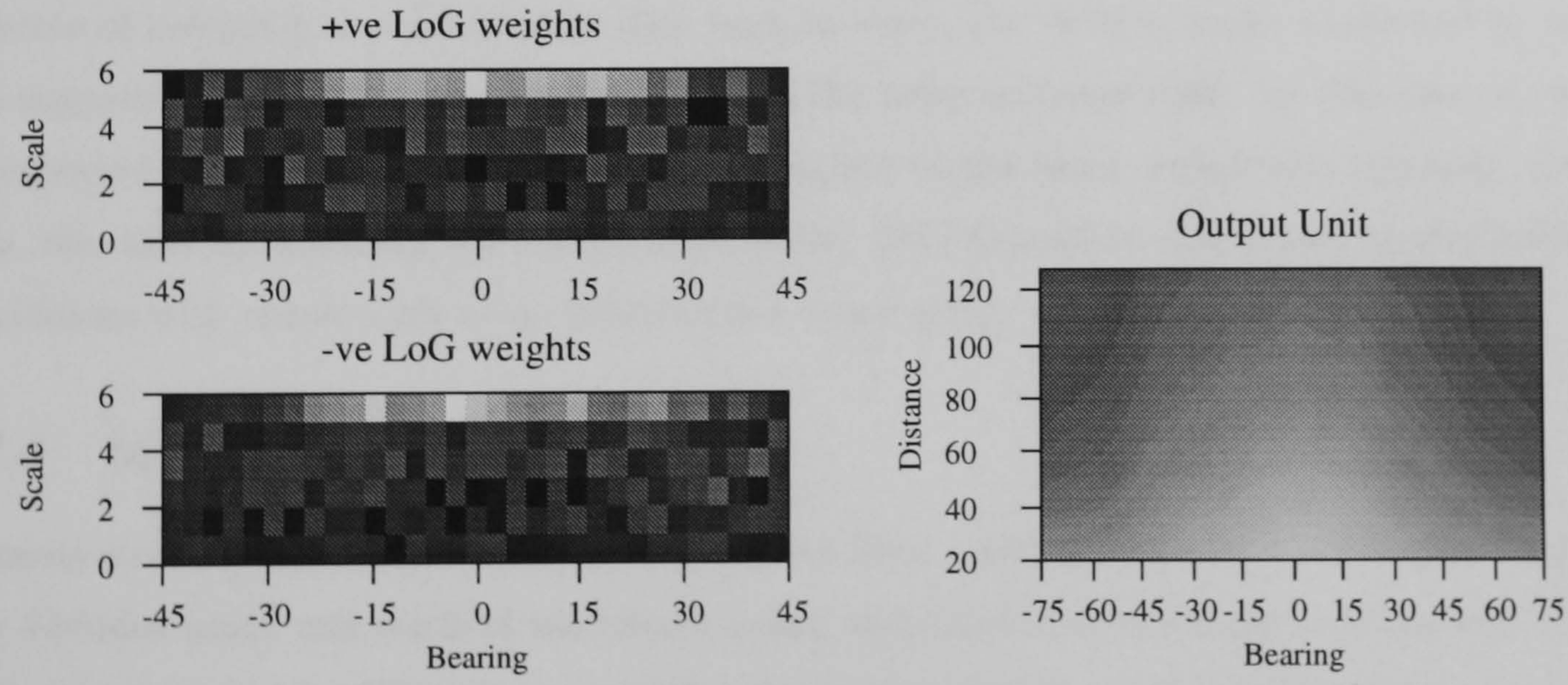


Figure 4.16: The learned weights and response patterns of direct rectified multiscale coding filter networks. Left column: Weights to the positive and negative multiscale filtered visual arrays. Right column: Activation of the output unit to the circle as a function of its distance and bearing relative to the center of the filter network's receptive field.
(For these plots: circle intensity = 0.7, wall intensity = 0.2.)

4.6.2 Results

Fig. 4.17 shows the performance after learning, together with the performance of the direct rectified multiscale coding animats from above. Without visual noise (fig. 4.17a), performance does not vary greatly with scale, although the medium scales do slightly outperform the finest and coarsest, and the rectified multiscale animats (note the different y-axis scale in fig. 4.17a). As explained above, direct rectified multiscale animats have positive weights concentrated at the coarsest 2 scales. Fig. 4.17a suggests that this may not be the optimal arrangement, though the difference is only around 0.03.

With independent sensory noise, there is a clear effect of scale (fig. 4.17b and c): up to scale 4, the coarser the LoG filter scale, the higher the performance of animats; and this effect is more pronounced at the 0.1 noise level. The two coarsest scales (4 and 5) lead to about the same performance in each noise condition, and this matches the performance of direct rectified multiscale coding animats.

4.6.3 Conclusion

These results support the hypothesis that with independent noise, the superior performance of rectified multiscale animats is due to the coarse scale LoG filter output, which enables adequate detection of the circle, whilst being relatively unaffected by independent noise.

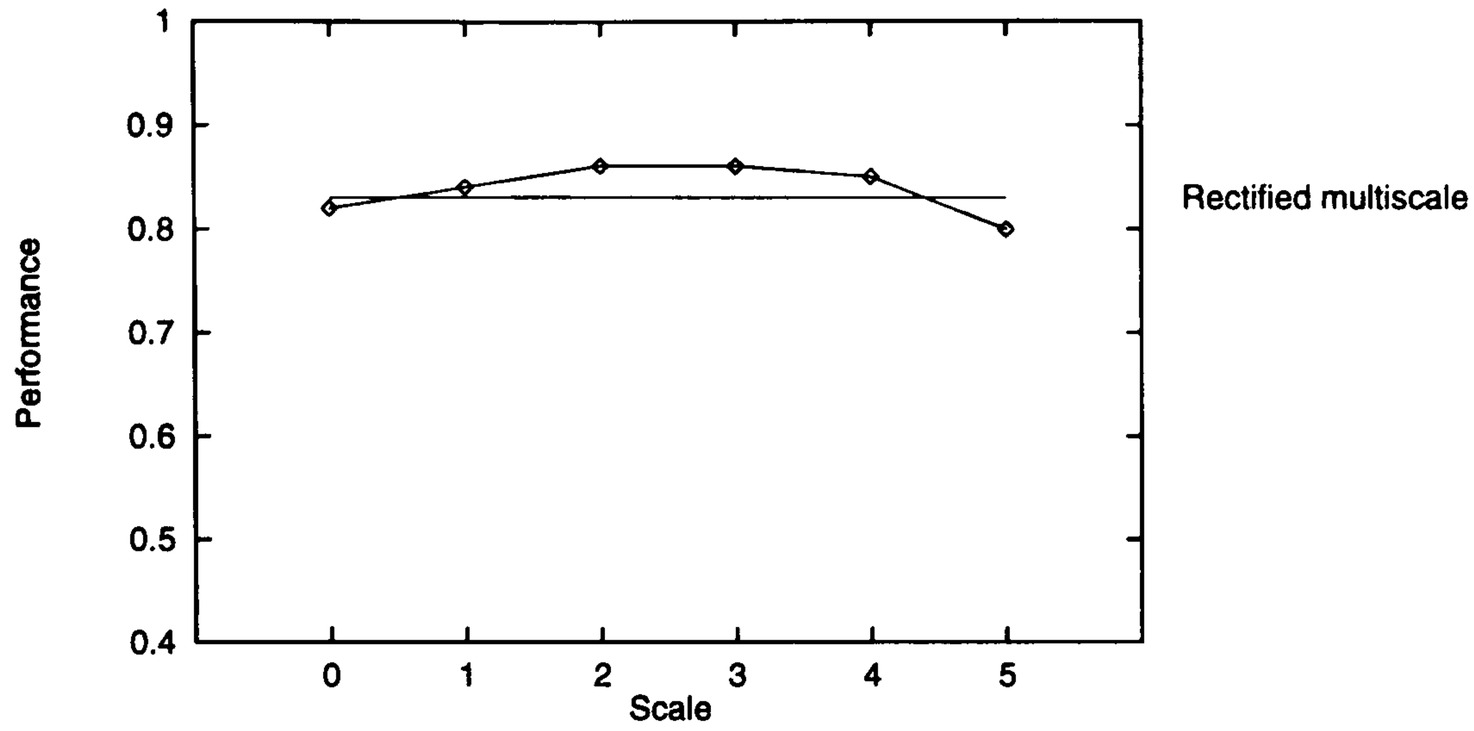
4.7 Coarse scale noise

Given that rectified multiscale coding animats outperform those using intensity coding in the presence of independent noise because they learn to detect the circle at scales unaffected by noise, this suggests an analogous result would occur if the noise is coarse scale. In this case, it would be expected that rectified multiscale coding animats would again outperform intensity coding ones, this time by detecting the circle at fine scales. This hypothesis was tested by the following simulations with coarse scale noise added to the visual array.

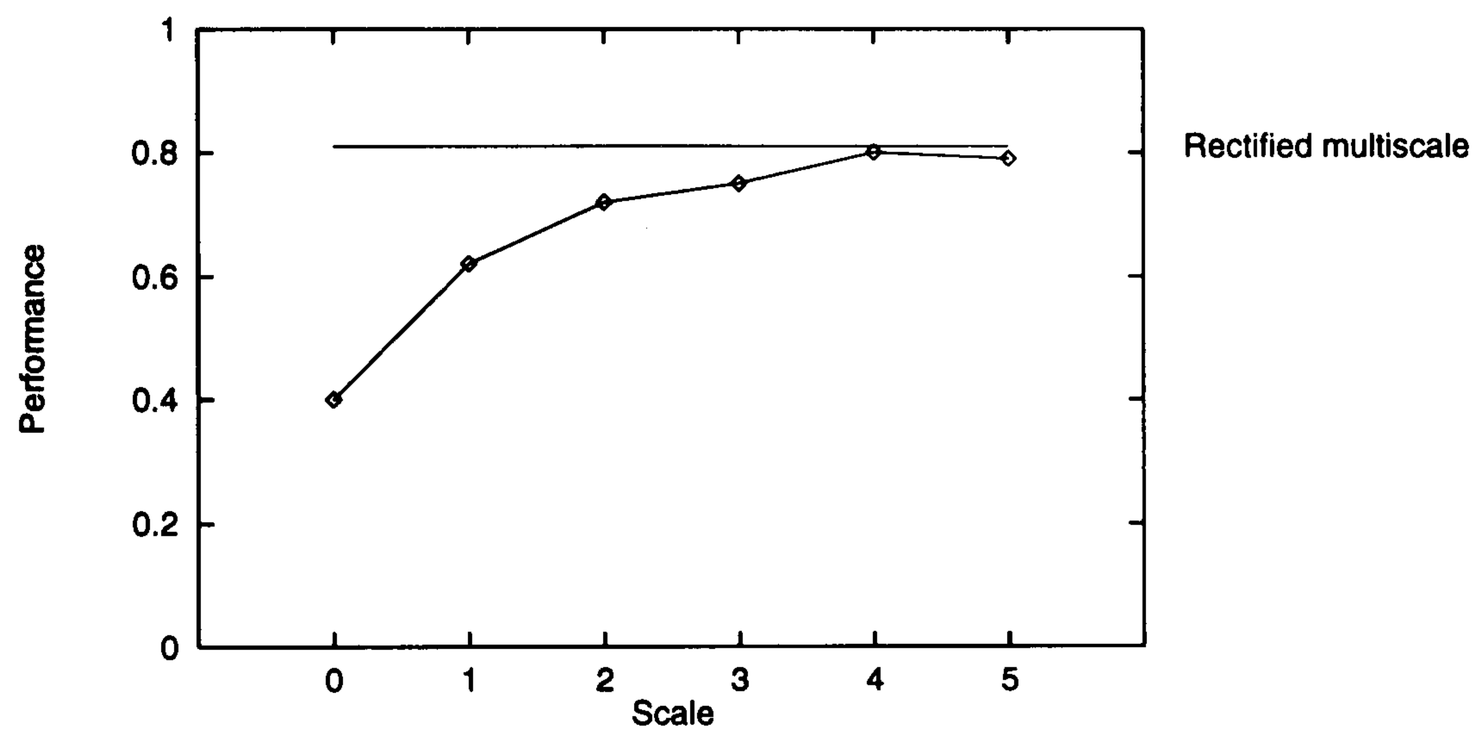
4.7.1 Method

Animats were simulated with intensity coding and filter networks with no hidden units (direct), 2 or 4 hidden units, and rectified multiscale coding and direct filter networks. Coarse scale visual noise was generated by filling an array with independent Gaussian noise of mean 0.0 and sd 0.33, and then convolving it with a Gaussian of standard deviation 3.0. This results in a coarse scale noise with standard deviation 0.1.

(a) Noise = 0.00.



(b) Noise = 0.05.



(c) Noise = 0.10.

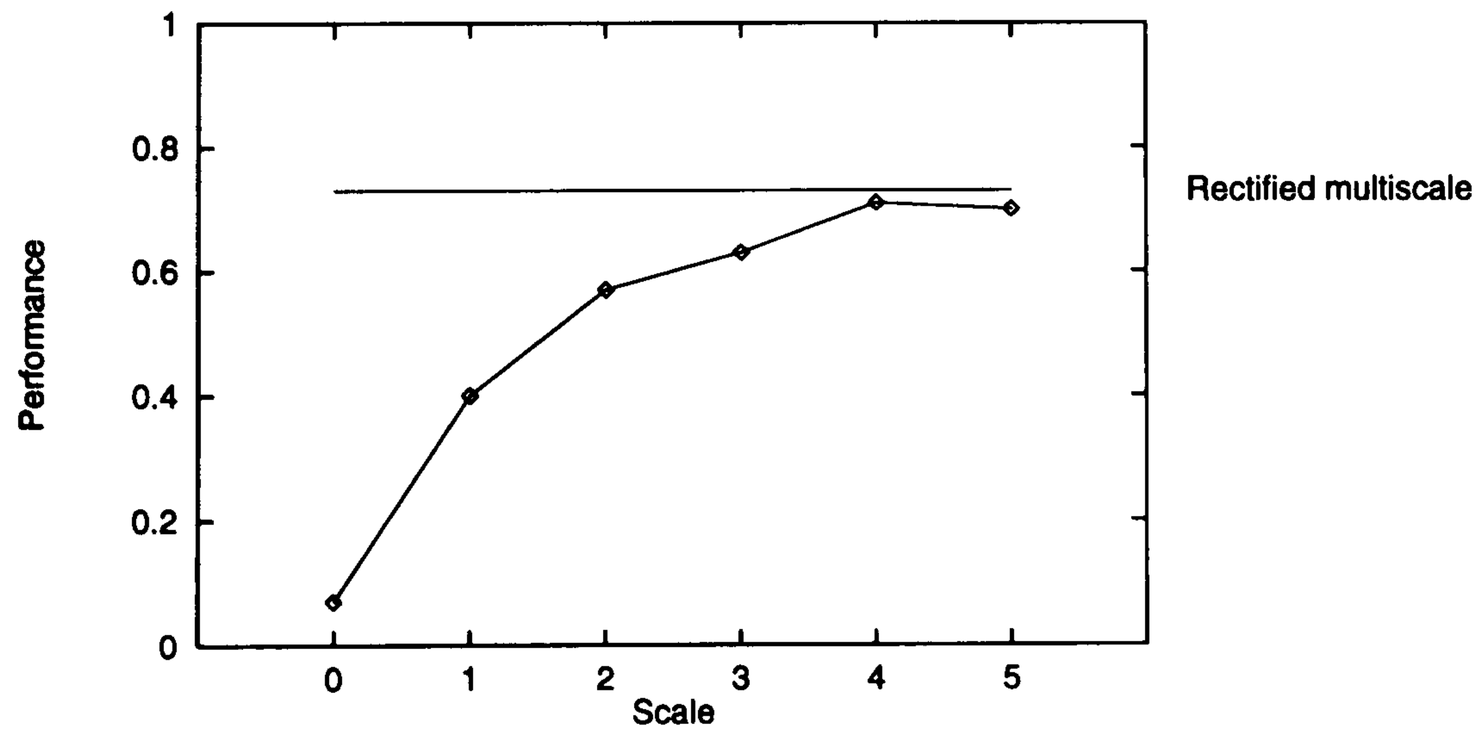
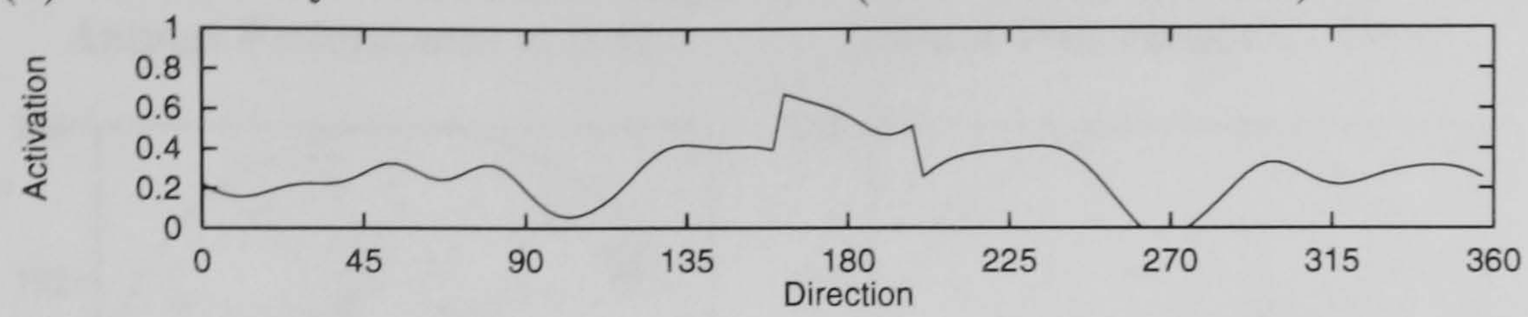


Figure 4.17: Mean performance of direct rectified single scale coding animats as a function of independent visual noise and scale. Scale 0 is the finest of the multiscales and scale 5 the coarsest. (a) Zero sensory noise. (b) Independent sensor noise of sd 0.05. (c) Independent sensor noise of sd 0.10. Each data point is the mean of three animats. Animats tested at the same noise level as during learning. For (a) and (b), all standard errors$\leq$0.01. For (c), all standard errors$\leq$0.02. Horizontal lines mark the performance level of direct rectified multiscale animats at the same noise level.

(a) Visual array with coarse scale noise. (Circle distance = 60)



(b) Visual array after multiscale filtering

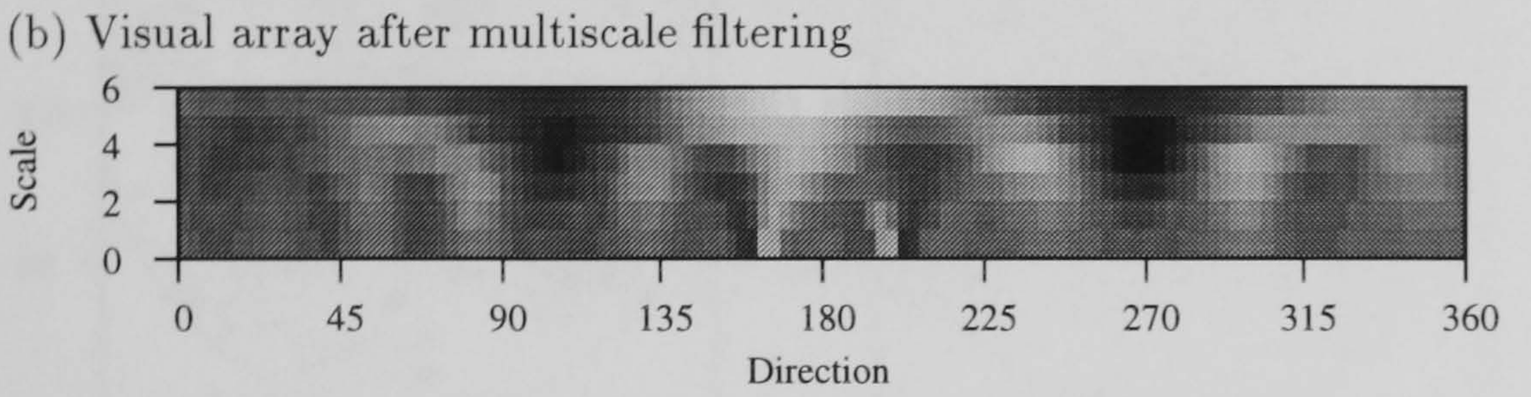


Figure 4.18: (a) Example visual array with coarse scale noise added.
(b) Multiscale convolution of the visual array. In contrast to the activity at the fine scales caused by independent noise, coarse scale noise causes activity at mostly the coarse spatial scales.

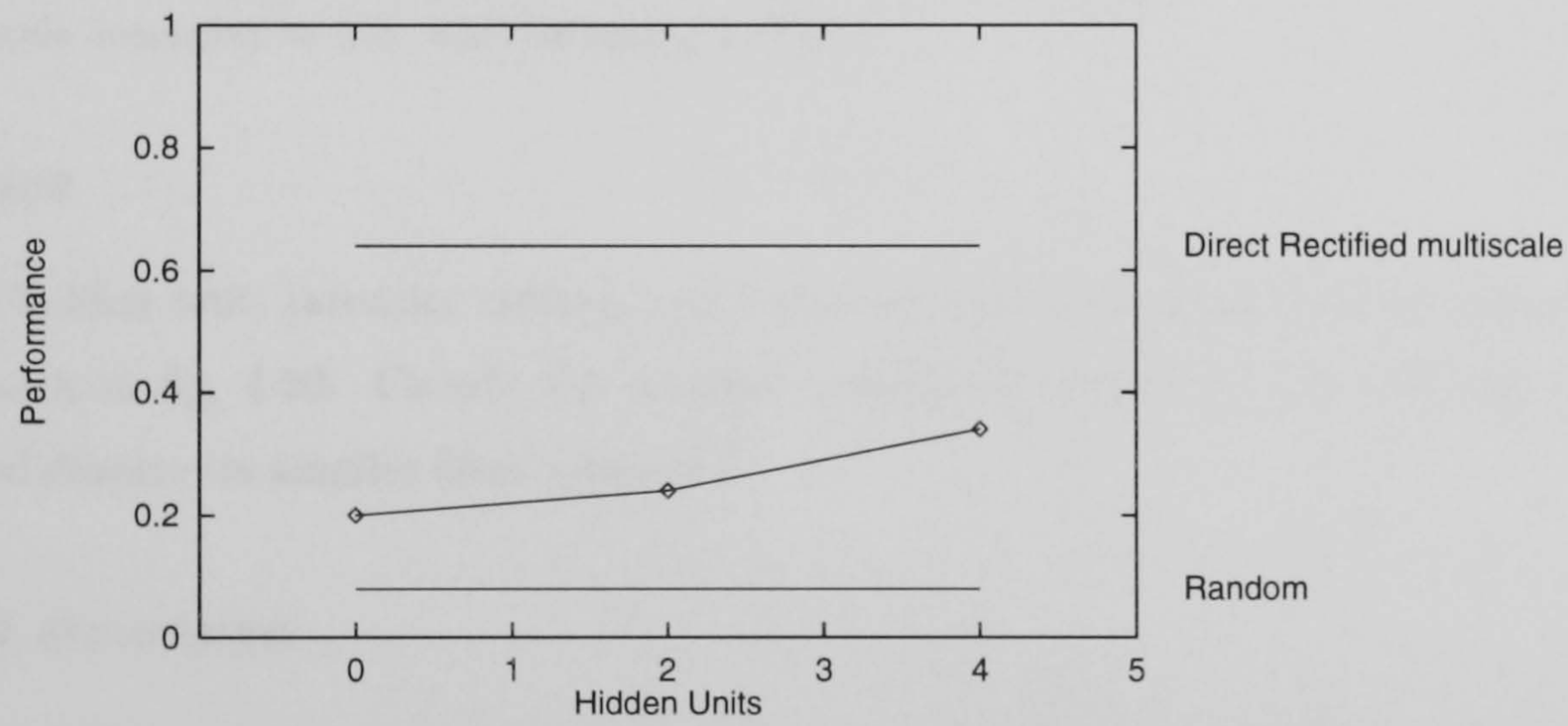


Figure 4.19: Performance of intensity and rectified multiscale coding animats after learning with coarse scale visual noise. Each data point is the mean of 3 simulations.
Standard error of direct rectified multiscale = 0.01; standard errors for intensity coding$\leq$0.04 for 0 and 2 hidden units, and 0.08 for 4 hidden units.

Fig. 4.18 shows a visual array with coarse scale noise added, and the result of multiscale filtering. Note that most of the activity due to the noise is confined to the coarse scales.

4.7.2 Performance

Fig. 4.19 shows the performance after 10k learning trials. Direct rectified multiscale coding animats perform at 0.64$\pm$0.01; direct intensity coding animats perform at 0.20, and this only increases to 0.34 with 4 hidden units. The difference between 4 hidden unit intensity coding and direct rectified multiscale coding is significant (t-test: t = 4.014, p<0.02). Not only does rectified multiscale filtering lead to significantly better performance, as was the case with independent noise, it is achieved with a smaller filter network.

(a) 4 hidden unit intensity coding.
Animat Performance = 0.44

(b) Direct rectified multiscale coding.
Animat Performance = 0.64

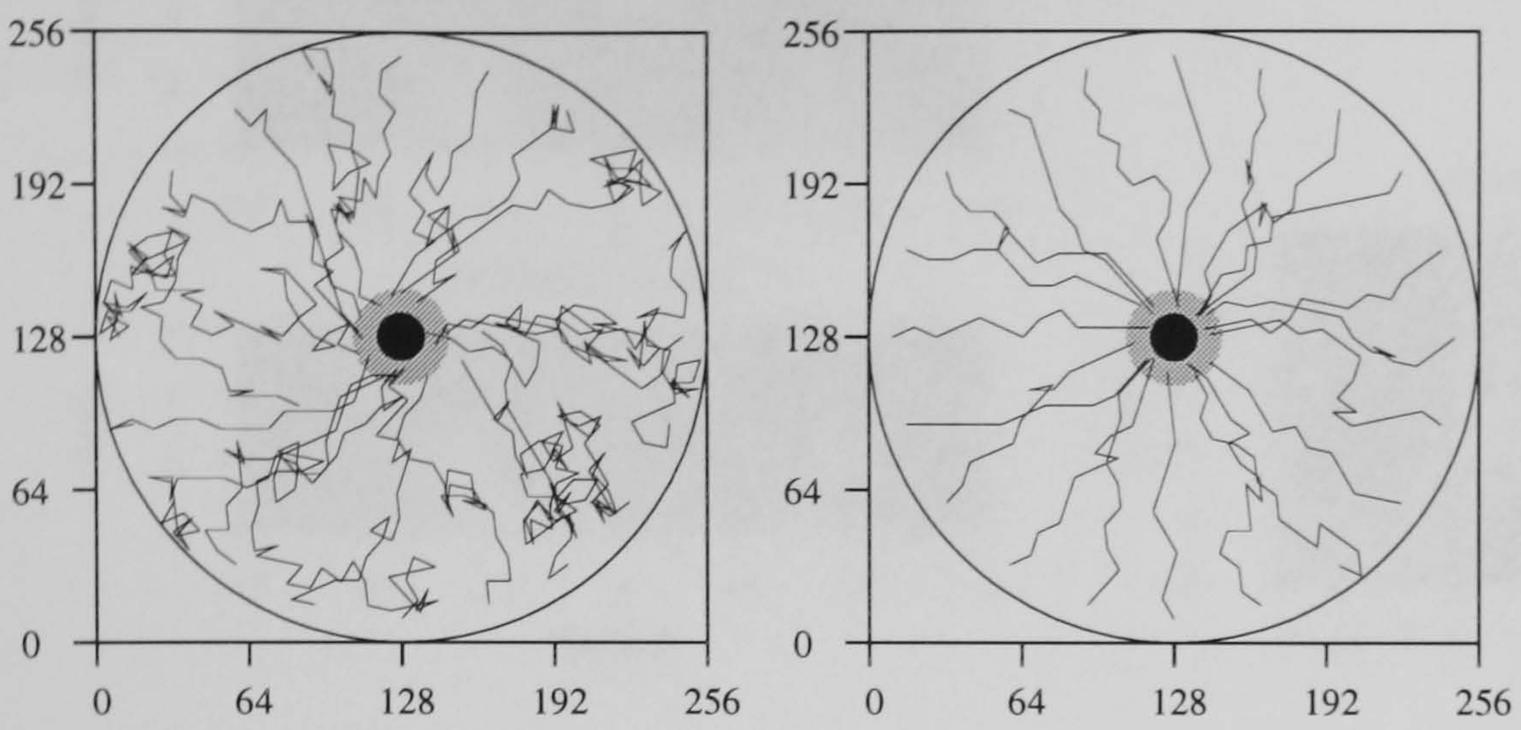


Figure 4.20: Typical paths of a 4 hidden unit, intensity coding animat, and a direct rectified multiscale coding animats learned (and tested) with coarse scale visual noise.
(For these plots: Circle intensity = 0.5, wall intensity = 0.2)

4.7.3 Behaviour

The behaviour of 4 hidden unit, intensity coding, and direct rectified multiscale coding animats after learning is shown in fig. 4.20. Clearly the rectified multiscale animat is more efficient at approaching the goal despite its smaller filter network.

4.7.4 Internal structure

Fig 4.21 shows the weights and response pattern of the direct rectified multiscale animat whose behaviour was shown above. The weights have a very different structure to that seen when learning with independent noise (fig. 4.21). Apart from a small patch in the center of the second from coarsest scale, large weights are concentrated in two patches at the fine scales. Each of these responds individually to the circle at far distance, giving rise to the two prongs of high output at distances greater than 60 shown in the right column. When the circle is closer, the weights respond to the activity caused by the two edges of the circle; individually to produce the outlying legs of high output and both together to produce the central region of high output when the circle is close.

Whereas in the independent noise case, rectified multiscale filter networks learn to detect the circle using the coarse scales, when the noise is coarse scale, the networks learn to detect the circle at the fine scales.

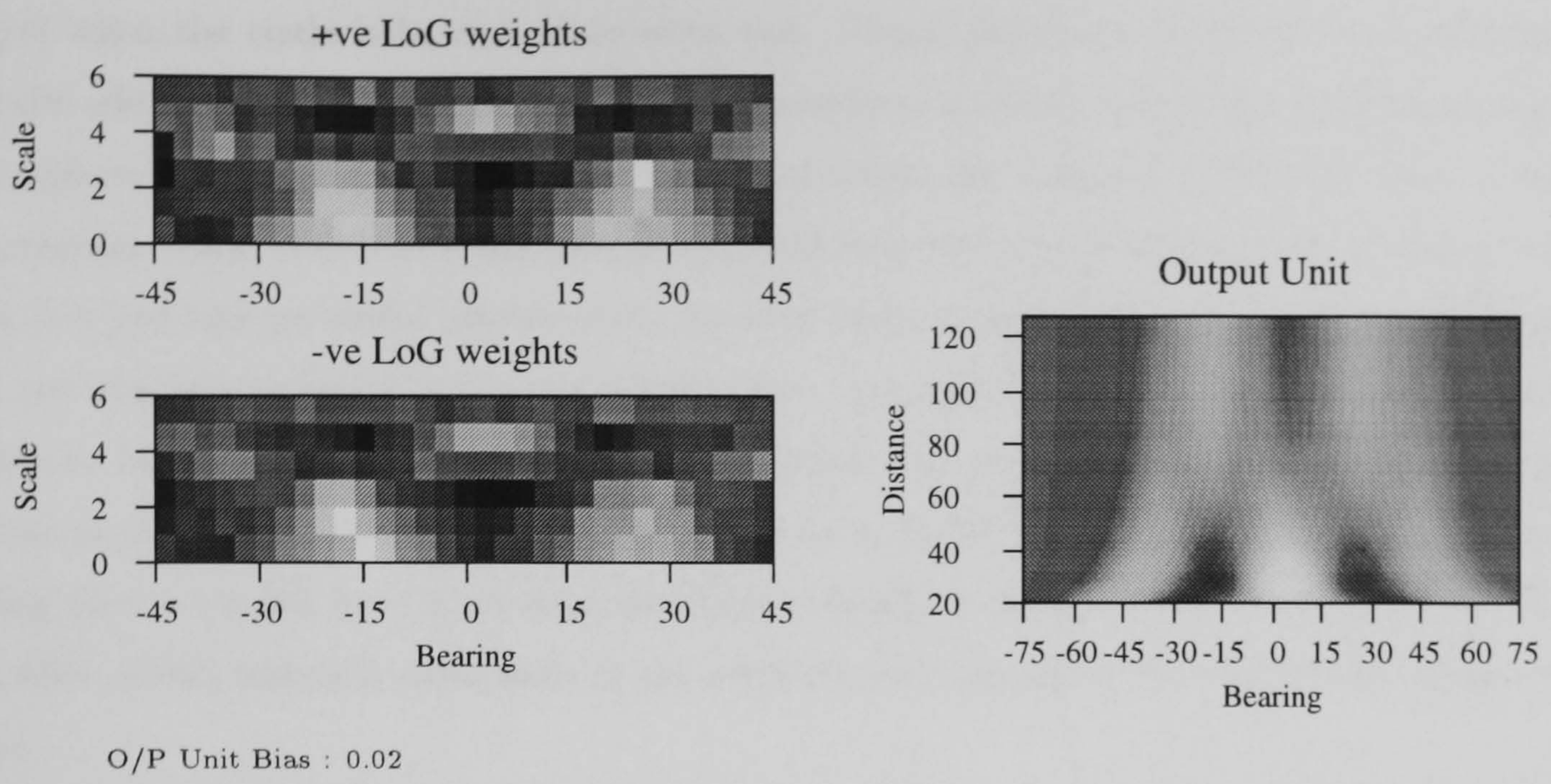


Figure 4.21: The weights and response profile of a direct, rectified multiscale coding animat after learning with coarse scale visual noise. The weights to the positive and negative multiscale filtered array are shown in the left 2 columns. Note how different these weights are to those after learning with independent noise. The right hand column shows the response of the unit to the circle at the distance and bearings given by the axes.
(For this plot: circle intensity = 0.7, wall intensity = 0.2, noise = 0.0.

4.7.5 Conclusion

These simulations have shown that with coarse scale noise, rectified multiscale coding animats learn to significantly outperform those with intensity coding because the fine scale LoG filter output enables adequate detection of the circle whilst being relatively unaffected by the noise. Direct rectified multiscale animats learn to exploit this to achieve efficient performance. Intensity coding animats with up to 4 hidden units are unable to learn to distinguish between the coarse scale noise and the image of the circle in the raw visual array.

4.8 Discussion

Animats in this chapter have learned to approach a solitary circle whose contrast varied randomly in sign and magnitude between trials. With the exception of the two unstable conditions, all animats learned to perform at a level far above chance. Without noise, direct animats learn to perform at the same level, regardless of coding. With independent visual noise, rectified multiscale coding animats significantly outperform intensity coding animats at both noise levels, and multiscale coding at the highest noise level. With coarse scale visual noise, direct rectified multiscale animats again significantly outperform intensity coding animats.

Because the task is to approach the circle, filter networks have to learn to produce a higher

output when the circle is in view than when not. Simple detection of the circle is sufficient to underlie efficient performance, and the learning problem can be viewed as discriminating between variation in the visual array due to the circle and irrelevant variation. Without noise, a source of irrelevant visual variation is the background wall intensity, which varies randomly from trial to trial, but provides no useful information. As with most animal vision, only intensity differences and not absolute intensity carry any information. Intensity coding animats become insensitive to this by learning a balanced weight structure which does not respond to a single valued image, regardless of its absolute intensity. In addition to being balanced, the learned weights of intensity coding filter networks have a coherent structure of positive weights on one side, and negative on the other, which responds maximally to the intensity step caused by the edge of the image of the circle.

Because the LoG filter is itself balanced, the multiscale array is only sensitive to contrast, and thus builds in a computation that intensity coding animats have to learn. As with most animal vision, only intensity differences and not absolute intensity carry any information. Intensity coding animats must learn the irrelevance of absolute intensity from experience with the visual array; multiscale coding animats have this built in.

With visual noise, the learning problem becomes discriminating variation in the visual array due to the circle from variation due to the noise. Multiscale filtering of the visual array facilitates this computation because the image of the circle causes activity at all the LoG scales used here, whereas independent noise causes activity at only the finer scales. Thus, multiscale filtering separates the circle and noise signals and the filter networks have only to learn to exploit this separation. Rectification of the multiscale array ensures that positive network weights can only add to output activation. So, positive weights can be concentrated on the coarse scales and activity at the fine scales ignored. Examination of the internal weight structure shows that direct rectified multiscale networks do learn this simple computation.

The finding that rectified single scale coding animats, with coarse scale input, learn to perform as well as the multiscale ones, whereas those with fine scale input perform poorly, further supports the case. With intensity coding, the circle and noise signals are not separated and so this computation must be performed by the filter network itself. This proved too hard a computational problem for them to learn to solve, even with 8 hidden units, and hence the relatively poor performance of intensity coding animats.

With coarse scale visual noise, the opposite situation occurs. The noise causes activity at mostly the coarse scales, whereas the circle can be detected at all scales. Direct rectified multiscale animats exploit this to achieve high performance by detecting the circle at the fine scales. Although it has not been simulated, it would be expected that with coarse scale visual noise, single scale animats with a fine scale filter would perform as well as those with multiscale filtering, whereas

those with a coarse scale filter would perform poorly.

Chapter 5

Learning visual subtended angle

5.1 Introduction

The environment of this chapter is the same as in the previous chapter: a solitary circle within an otherwise empty arena. In this chapter, the radius of the circle varies randomly between trials, and as in the last chapter, contrast varies randomly between trials. In the previous chapter, animats received reinforcement when they moved close to the circle. Here, animats of the same design receive reinforcement when they move to where the visual angle subtended by the circle falls within a goal range.

These simulations are motivated by Cartwright and Collett's (1983) behavioural experiments of insect learning which are described first.

5.1.1 Insect learning of subtended angle

Cartwright and Collett (1983) trained bees to locate a food source at a fixed distance and direction from a single featureless cylinder, in an otherwise empty arena. The bee's environment was impoverished so that as far as possible, the visual appearance of the featureless cylinder was the only sensory information available to guide the bees toward the goal. Between trials, the landmark and food were translated (but not rotated), to further ensure that the insects could only rely on their vision, rather than path integration. Using vision to learn to move to the location of a food source is part of the everyday life of honeybees, and this is reflected in their successful learning of Cartwright and Collett's (1983) task. After a number of trials, bees fly directly to the food source. The bees are tested by recording where they search in trials with no food source, under the assumption that this reflects where they expect to find the food. In such test trials, especially with environments modified from that experienced during learning, the pattern and location of search can provide information about the computations underlying the bee's behaviour.

Learning a goal direction from a featureless cylinder implies the use of a non-visual direction sense, and further experiments have demonstrated the use of a magnetic sense in honeybees (Collett and Baron, 1994). The simulations in this chapter focus upon the purely visual input, and following Cartwright and Collett (1983), concentrate upon how the animats learn to move to the correct distance. In chapter 7, a simple compass sense is added to the animats, which allows simulation of goals defined by both distance and direction. This incremental approach permits a clearer and more detailed understanding of animats and is usual within computational neuroethology (eg Cliff et al, 1996).

To learn a goal distance from the cylinder requires that the bees can compute some relevant information from its image. Even in this relatively simple case, the spatiotemporal image of the featureless cylinder contains a number of different cues that could be used to guide movement. Testing the bee with cylinders of a different radius to that used during learning, and recording their search distribution as a function of distance permits further specification of what the bees are computing. When tested with a cylinder of half the radius, bees search closer to the cylinder, and when tested with a cylinder of twice the radius they search further away. Cartwright and Collett (1983), compared the bee's search distributions in the test trials with those predicted by assuming that the bee computes one of the following three quantities, all of which would enable the bee to locate the goal, but predict different search distributions in the altered radius test trials.

Firstly, bee's could compute and store the retinal angle subtended by the cylinder as seen from the goal. To locate the goal, the bee then flies to where the subtended angle of the cylinder matches the stored subtended angle. Secondly, bees could compute the velocity at which the image of the cylinder translates across the retina and store the retinal velocity at the goal (motion parallax). Given a known or constant flying velocity, the bee then fly to where the retinal velocity matched that stored. Thirdly, bee's could compute the rate of change of the angle subtended by the image of the cylinder as it approached the goal (looming). Each of the three alternatives predicts different search distributions for the tests with different sized cylinders to those experienced during learning. Subtended angle closely matched the bee's search distribution, with the alternative hypotheses producing discountable distributions. Thus Cartwright and Collett (1983, p 527) conclude "that bees learn the angular size of the landmark when viewed from the position of the food source and use this to guide their return."

Other experiments such as Cheng et al (1987) and Srinivasan et al (1989) have shown that bees are able to use other cues such as motion parallax and texture to estimate distance in different situations. Subtended angle is one of a number of cues that bees use to estimate distance. It forms the basis for the simulations in this chapter, where convolution animats learn to move to locations where the angle subtended by a solitary circle falls within a certain range.

Cartwright and Collett (1983) then changed the cylindrical landmark for a large black, filled-in

square. Again bees learned the angle subtended by the square to guide search. When tested with just the frames of the square, bees show a very similar search distribution to that with the solid square. However, Cartwright and Collett (1983) note that: "Although the [search] distributions were unchanged, it was nonetheless clear that the bee had noticed the difference between the solid board and the frames. It was much less eager to fly and search when tested with frames than it was when the landmark was solid." (p 528). This is an important result in terms of differentiating models of the computations underlying the bee's behaviour. Not only must a candidate model be able to learn to search where the angle subtended matches, but it must generalise like the bees, and be relatively unaffected by replacing a solid subtended angle by just its edges.

5.2 Simulations

Convolution animats, identical in both processing and learning to those of the last chapter, here learn to move to regions where the angle subtended by a circle falls within a goal range. The simulations in this chapter will show that subtended angle can be reactively learned both with, and without, the bee-like generalisation to edges only. Because the animats are reactive, and the circle featureless, subtended angle is the only cue available to the animats to judge distance. Therefore, finding that animats learn this cue is not very interesting, except from a computational point of view. That bees learn subtended angle to estimate distance is an interesting result because, as explained above, there are many cues they could use. The interesting question for animats is whether, after leaning solid subtended angle, they generalise like the bees when tested with the edges-only circle.

The simulated environment is a circular arena of radius 128, containing a solitary circle whose radius varied randomly between 10 and 30 between trials (fig. 5.1). As in the last chapter, the circle and wall intensities are chosen randomly and independently from between 0.0 and 1.0 for each trial. Animats receive reinforcement of 1.0 when they move to where the angle subtended by the circle subtends between 15 and 21 degrees, and 0.0 reinforcement elsewhere.

In order to approach the circle, as in the previous chapter, animats only have to learn to detect it; the network must learn a filter shape that outputs a higher value when the circle is in view than when not. The present task is more difficult because it requires the network to learn to produce different outputs depending upon the angle subtended by the circle. If the angle subtended by the circle is too small, animats must move toward it; if it is too large, they must move away.

Varying the circle's radius between trials results in the number of steps to goal being unpredictable from the visual input. Hence a stable utility function does not exist. This may be expected to make the reinforcement learning of a utility function more difficult, but it is not particularly focussed upon in the rest of this chapter.

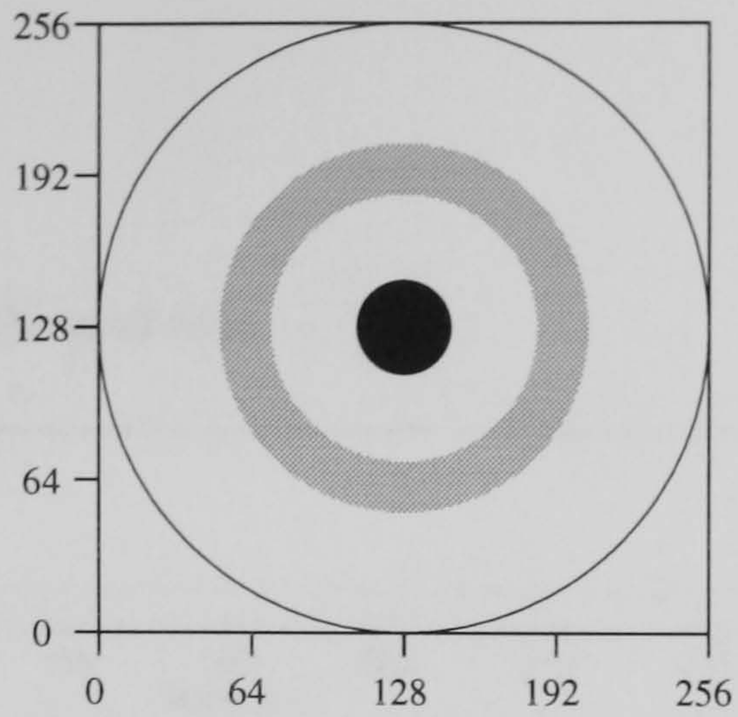


Figure 5.1: The environment is a circular arena of radius 128, empty, except for a solitary circle of radius varying between 10 and 30. The invisible goal region, shown in gray, is where the angle subtended by the circle is between 15 and 21 degrees. Animats receive reinforcement of 1.0 when entering this zone, and zero reinforcement elsewhere.

5.2.1 Visual array

The 120 element, 1-dimensional visual array is a rough approximation to a horizontal slice through the bee's 2-dimensional visual image of a featureless cylinder in a featureless room. Though the number of elements is about the same, a major difference is that the bee's receptors are not uniformly spaced across the retina (Fransechini et al, 1992). Fig. 5.2 shows example visual arrays from outside, within, and inside the goal ring, together with their multiscale convolutions. As the angle subtended by the circle increases, the *row* of maximum activation alters within the multiscale array. When the circle subtends around the goal angle, activity is maximal at scales 3 and 4; when the circle subtends a larger angle, activity is maximal at coarser scales; when the circle subtends a smaller angle, activity is maximal at the finer scales. The maximally active row is independent of contrast. Thus multiscale filtering makes visual subtended angle more explicit in that subtended angle, regardless of contrast, is transformed into an ordered position within the multiscale array.

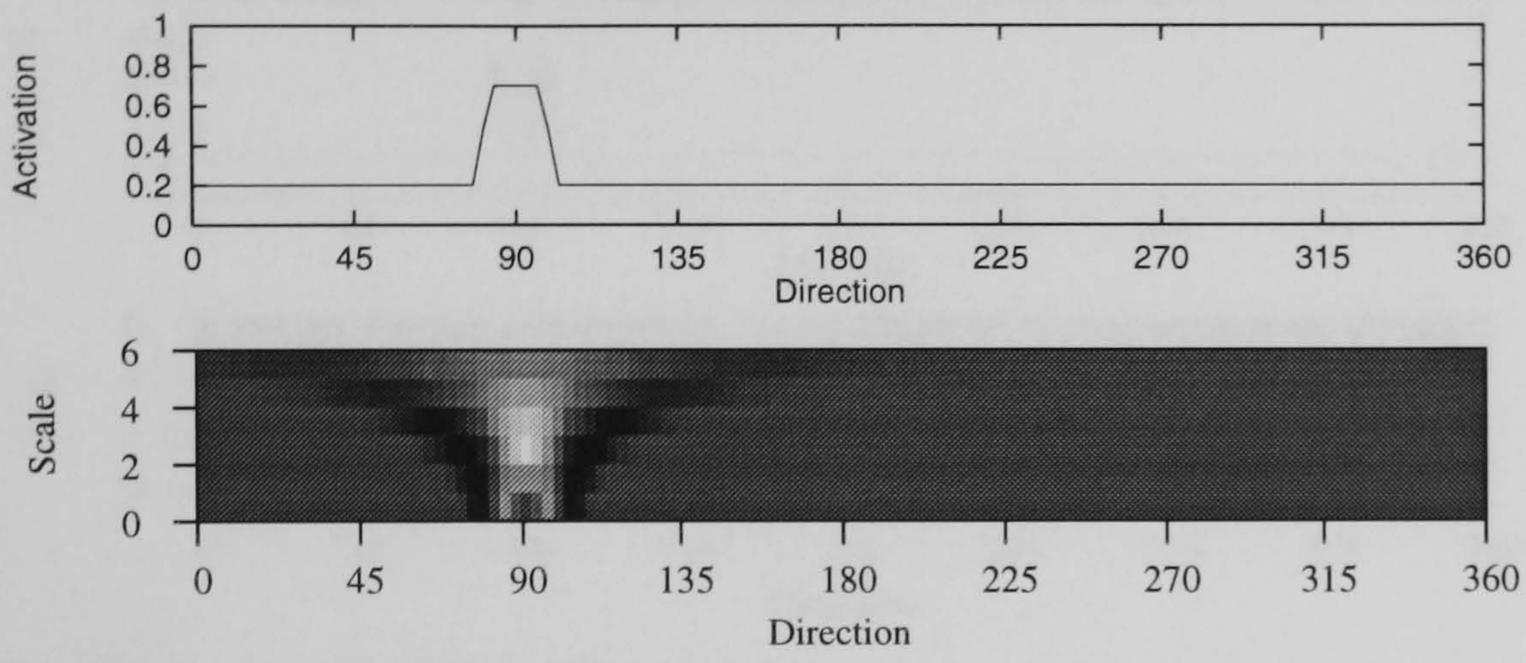
In order to simulate the edge-only tests, all elements within the solid patch of circle intensity are set to the background, wall intensity, level. Fig. 5.3 shows the edge only visual array for the circle at the same distances as in fig. 5.3. Except at far distance, activation in the multiscale array is confined to the fine scales. Unlike the solid case, with edges-only, the scale of maximal activation does not vary regularly with subtended angle.

5.3 Simulation method

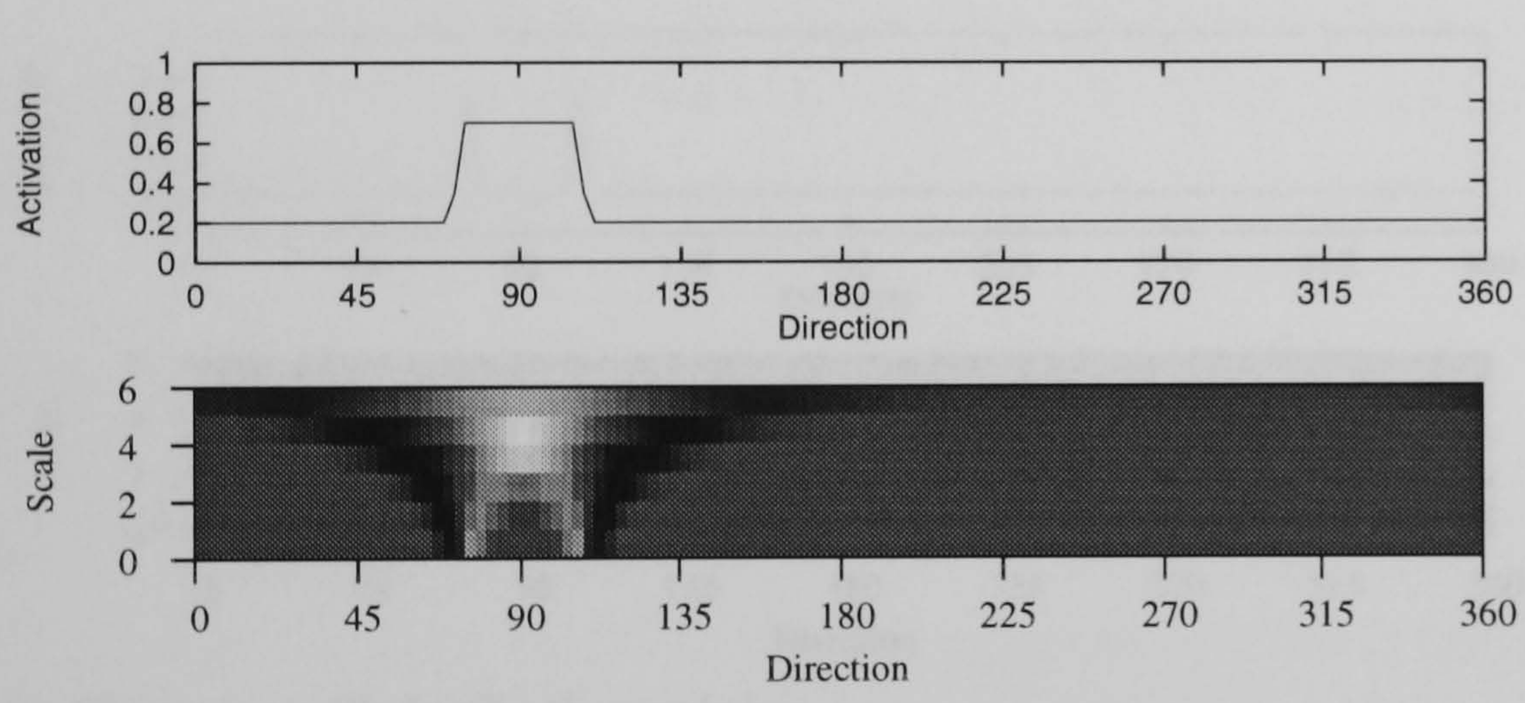
Except for the change of goal region, and the varying circle radius, the environment, and animats remain the same as in the previous chapter.

The 120 element visual array is coded as either a 1 dimensional intensity array, or convolved

(a) Distance = 125. Outside the goal ring.



(b) Distance = 67. Within the goal ring.



(c) Distance = 40. Inside the goal ring.

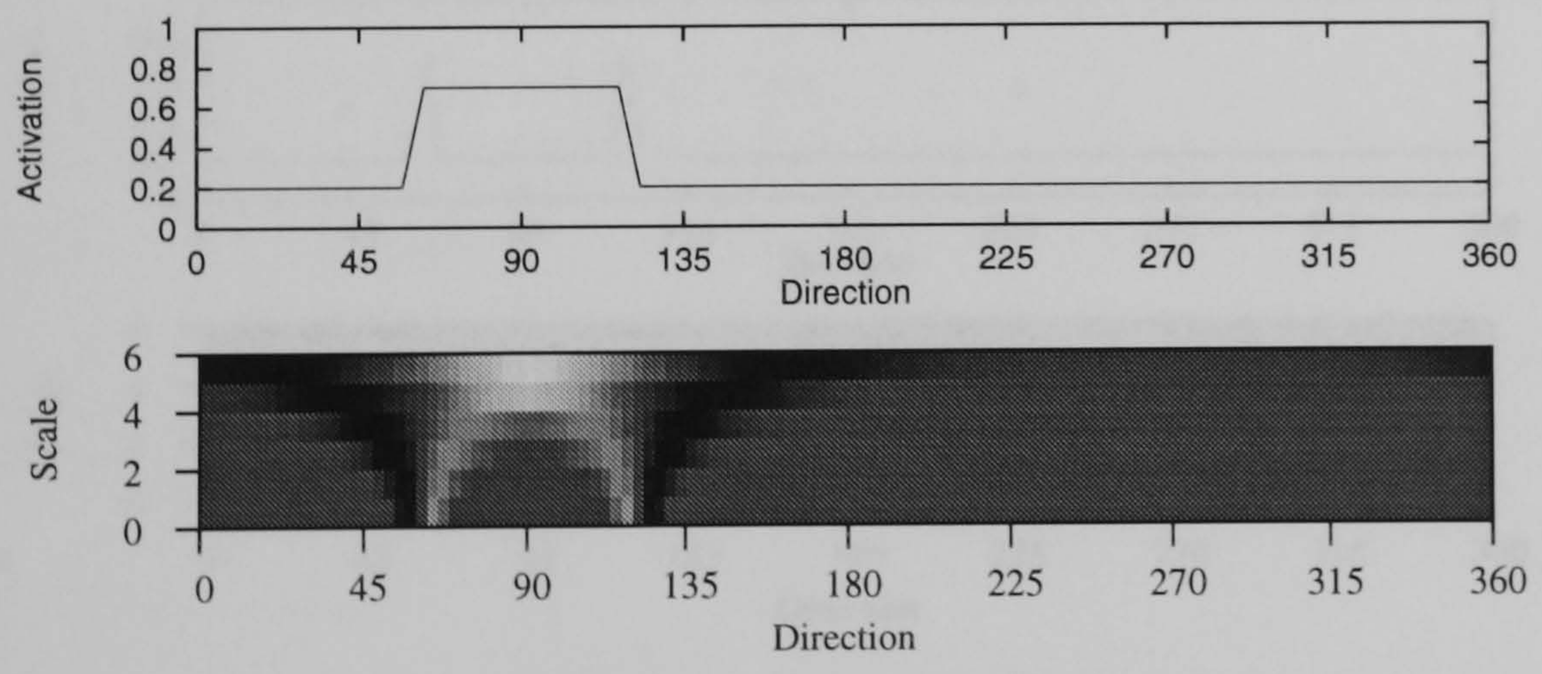
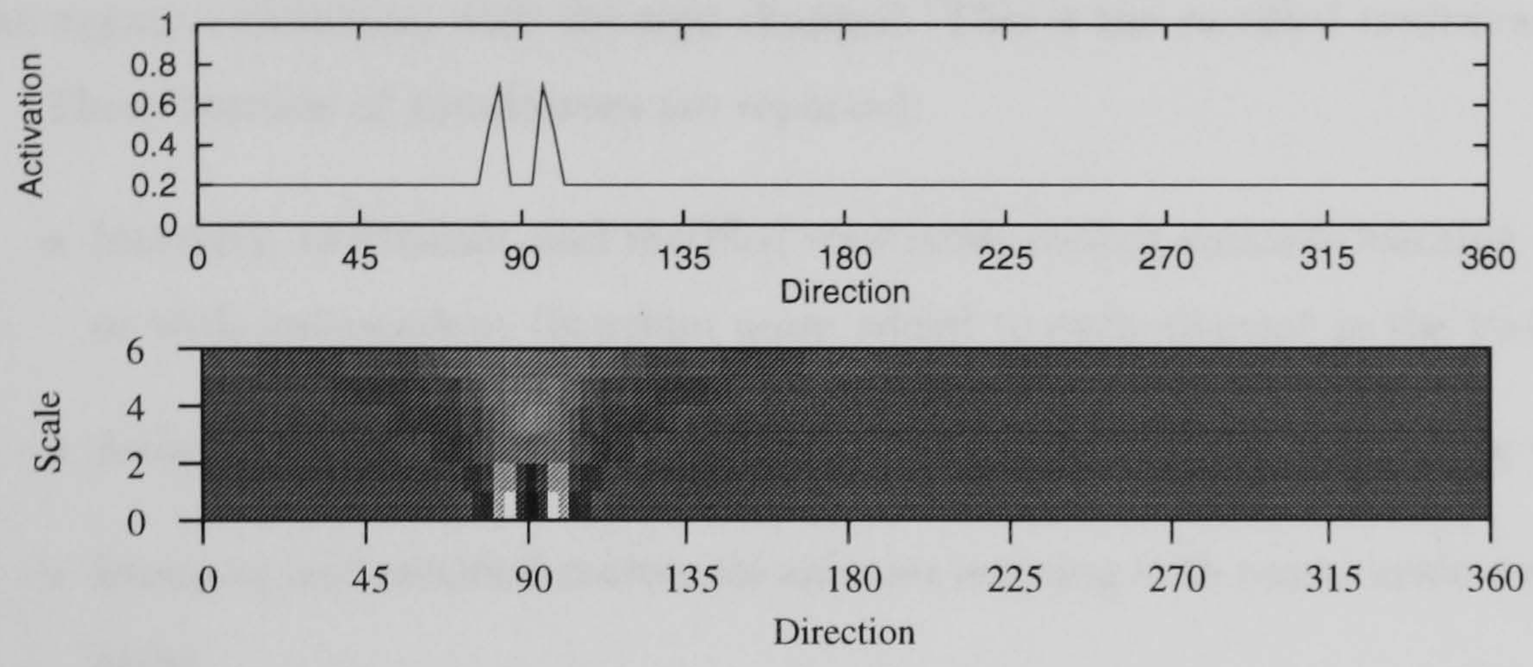
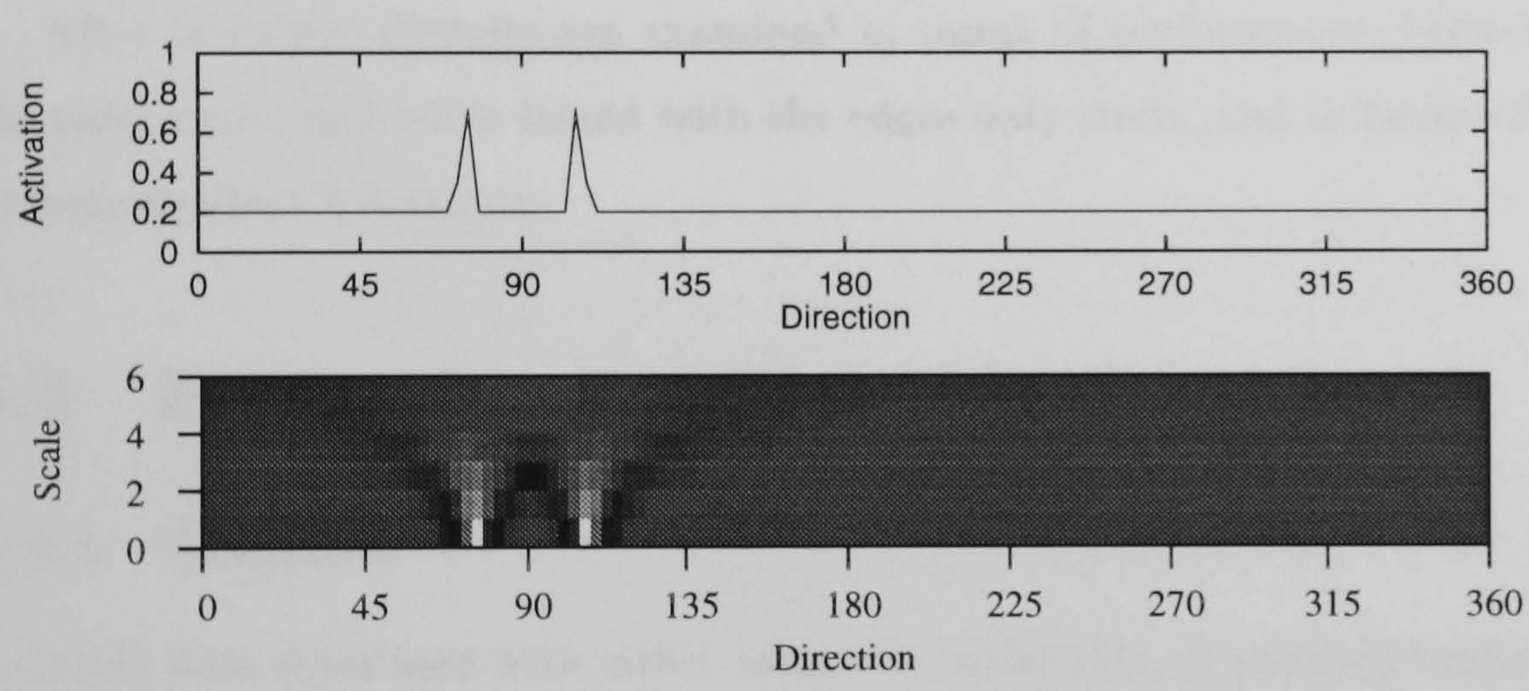


Figure 5.2: The visual image of the solid circle, intensity coded and after multiscale convolution. (Circle radius = 20, circle intensity = 0.7, wall intensity = 0.2, noise = 0.0)

(a) Distance = 125. Outside the goal ring.



(b) Distance = 67. Within the goal ring.



(c) Distance = 40. Inside the goal ring.

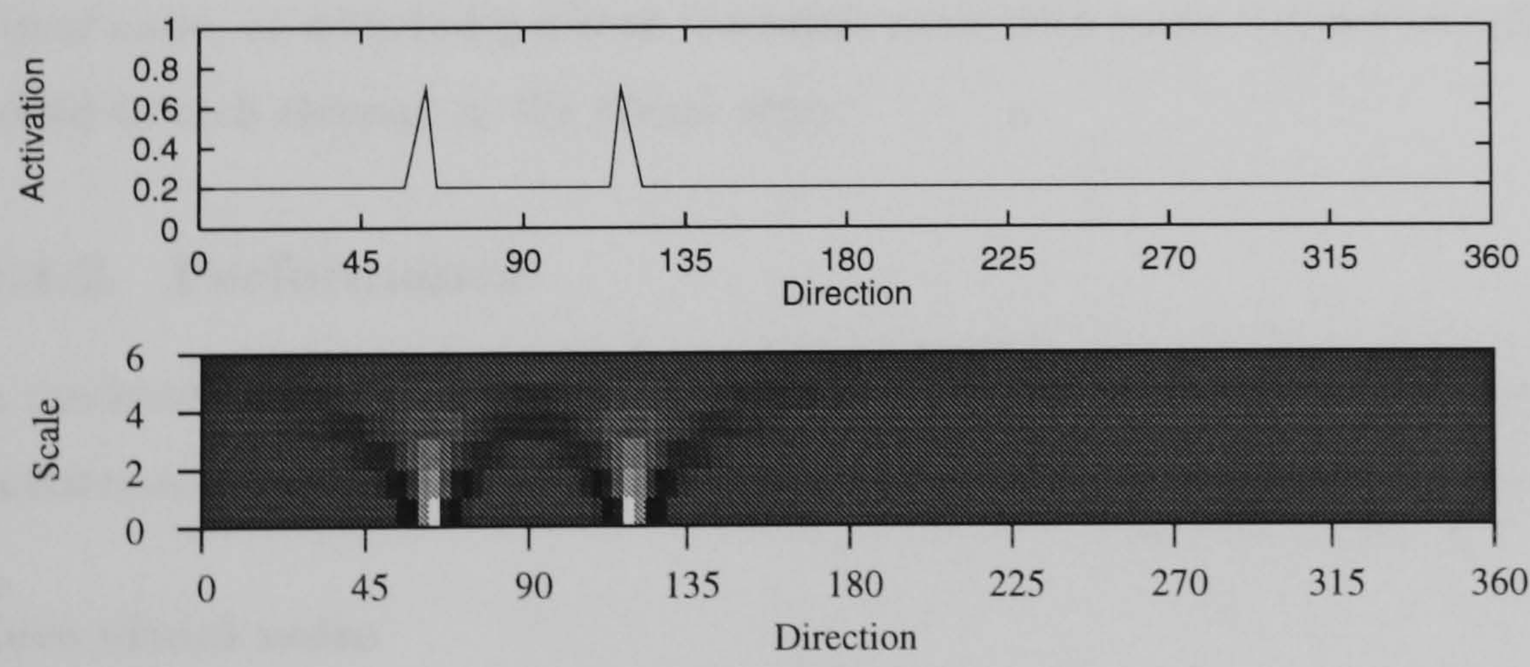


Figure 5.3: The visual image of the edge-only circle, intensity coded and after multiscale convolution. All elements within the image of the solid circle are set to the wall intensity level. (Circle radius = 20, circle intensity = 0.7, wall intensity = 0.2,noise = 0.0)

with 6 LoG filters (of the same scales as in the previous chapter), to produce a 6×120, 2-dimensional array (fig. 5.2). The 2-dimensional multiscale filtered array then becomes input to the animat as it is, the multiscale coding case; or, is split into two separate arrays, one containing only the positive elements in the multiscale array (negative elements turned to zero), and the other containing only the negative elements, with the sign changed. This is the rectified multiscale coding case.

Three batches of simulations are reported:

- Intensity, multiscale, and rectified multiscale coding animats learning with zero visual noise, or with independent Gaussian noise added to each element in the visual array.
- Animats learning from a single scale of the rectified multiscale array with zero noise.
- Intensity and rectified multiscale animats learning with coarse scale noise added to the visual array.

After learning, animats are examined in terms of performance, behaviour when tested with the solid circle, and when tested with the edges-only circle, and in terms of the internal structure underlying their behaviour.

5.4 Zero and independent visual noise

5.4.1 Method

Animats were simulated with either intensity, multiscale, or rectified multiscale coding, and filter networks with no hidden units (direct), or 2,4, 8,, or 12 hidden units. Animats learned with no visual noise, or with Independent Gaussian noise with mean 0.0 and standard deviation (sd) 0.10 added to each element in the visual array.

5.4.2 Performance

A randomly behaving animat, tested in the same conditions as learning animats (1000 trials of a maximum of 500 steps, starting from random locations), has a mean performance of 0.23 ± 0.01.

Zero visual noise

Learning proceeded for 50K trials for multiscale and rectified multiscale coding animats; 50k for direct and 2 hidden unit intensity coding animats, and 100k trials for 4 to 12 hidden unit, intensity coding animats. These learning times were sufficient for convergence in each condition. No stability problems were found in any condition. Learning curves for the 2 hidden unit condition are shown in fig. 5.4; these are typical.

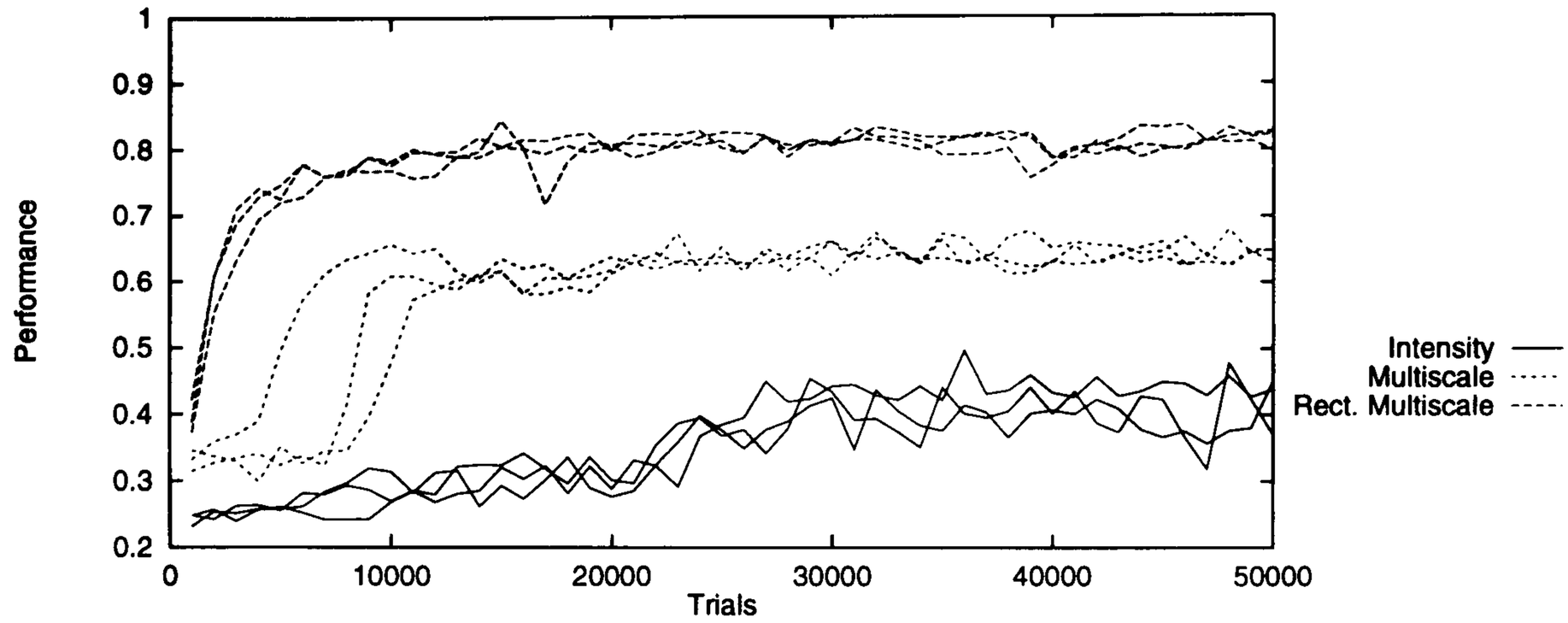


Figure 5.4: Learning curves for 2 hidden unit, zero sensory noise, condition in intensity, multiscale and rectified multiscale coding conditions. In each condition, 3 learning curves are shown. Each data point is the mean performance for the animat over the previous 1000 trials.

Fig. 5.5 plots the performance of the animats after learning. Each animat was tested over 1000 trials without learning. Also shown for comparison is the performance level of the randomly behaving animat.

Direct intensity coding animats perform little better than the random animat. With two hidden units, this rises to 0.4; performance then increases with hidden unit size up to 8 hidden units which perform at 0.75. No further performance increase is found with 12 hidden units.

Direct, multiscale coding animats perform at 0.43, and this increases to 0.65 with two hidden units. Performance then increases slightly with size. With 8 hidden units, multiscale coding animats perform at around the same level as those with intensity coding.

Direct, rectified multiscale coding animats, perform at 0.73, matching the best of the animats with intensity and mulitscale coding. With two hidden units this increases to 0.82, a significant increase (t-test: $t=4.53$, $p<0.02$), reflecting differences in what the two sizes of network have learned that are examined in section 5.4.4. This performance is not exceeded with more hidden units.

The performance of 2 hidden unit, rectified multiscale animats is significantly better than the best of intensity coding (t-test: 12 hidden unit intensity vs 2 hidden unit rectified multiscale: $t=3.34$, $p<0.05$). With multiscale coding, the difference approaches significance (t-test: 8 hidden unit multiscale vs 2 hidden unit rectified multiscale: $t=2.76$; for $p<0.05$ a t of greater than or equal to 2.78 is required). Hence, a further learning run of 3 animats was simulated in both conditions. The resulting animats performed at similar levels to the original ones (see appendix). Combining the scores, rectified multiscale coding leads to significantly better performance than multiscale coding (t-test: 8 hidden unit multiscale vs 2 hidden unit rectified multiscale, $n=6$: $t=5.23$, $p<0.01$), while requiring fewer units.

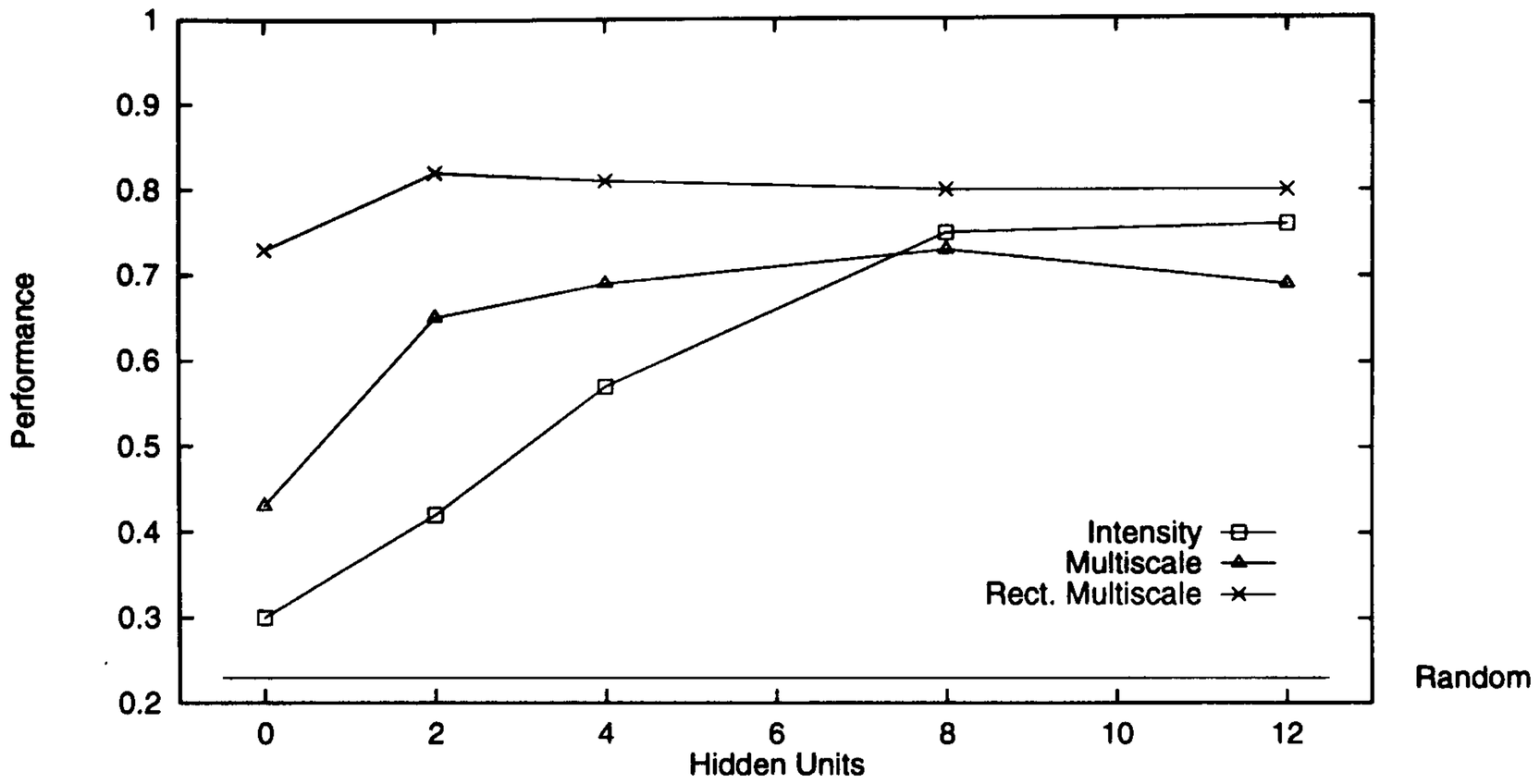


Figure 5.5: Performance as a function of network size for intensity, multiscale and rectified multiscale coding learning with zero noise. Each data point is the mean performance of 3 animats. All standard errors $\leq$0.05 for intensity coding, and $\leq$0.02 for multiscale and rectified multiscale coding.

Independent visual noise

Fig. 5.6 shows the performance after convergence of animats learning with independent Gaussian noise of mean 0.0 and sd of 0.1 independently added to each visual array element. Learning proceeded for 50K trials for rectified multiscale coding animats and 100k for intensity coding animats. Multiscale coding animats were not simulated in this condition.

Intensity coding animats with 4, or less, hidden units perform near randomly. With 8 or 12 hidden units, performance is around 0.57, a 0.2 drop on their noiseless performance above. Direct rectified multiscale coding animats also perform at 0.56. With 2 hidden units, performance rises to 0.68, a significant increase (t-test: Direct vs 2 hidden unit rectified multiscale coding: t=13.23, p<0.001), and this is not exceeded with more hidden units. The difference between rectified multiscale and intensity coding is significant (t-test: 2 hidden unit rectified multiscale vs 8 hidden unit intensity coding: t=3.45, p<0.05). As in the noiseless case, the superior performance of rectified multiscale filtering animats is achieved with fewer units.

5.4.3 Behaviour

Intensity coding

Fig. 5.7 shows example paths for intensity coding animats of increasing network size in the zero noise condition. Direct animats drift toward the circle when outside the goal ring; when inside it, animats just bounce around the circle. With 2 hidden units, animats move rapidly to the goal from outside the ring, but have not learned to move outward when within the ring. With 4 hidden units, the situation is improved somewhat, and with eight hidden units, animats drift

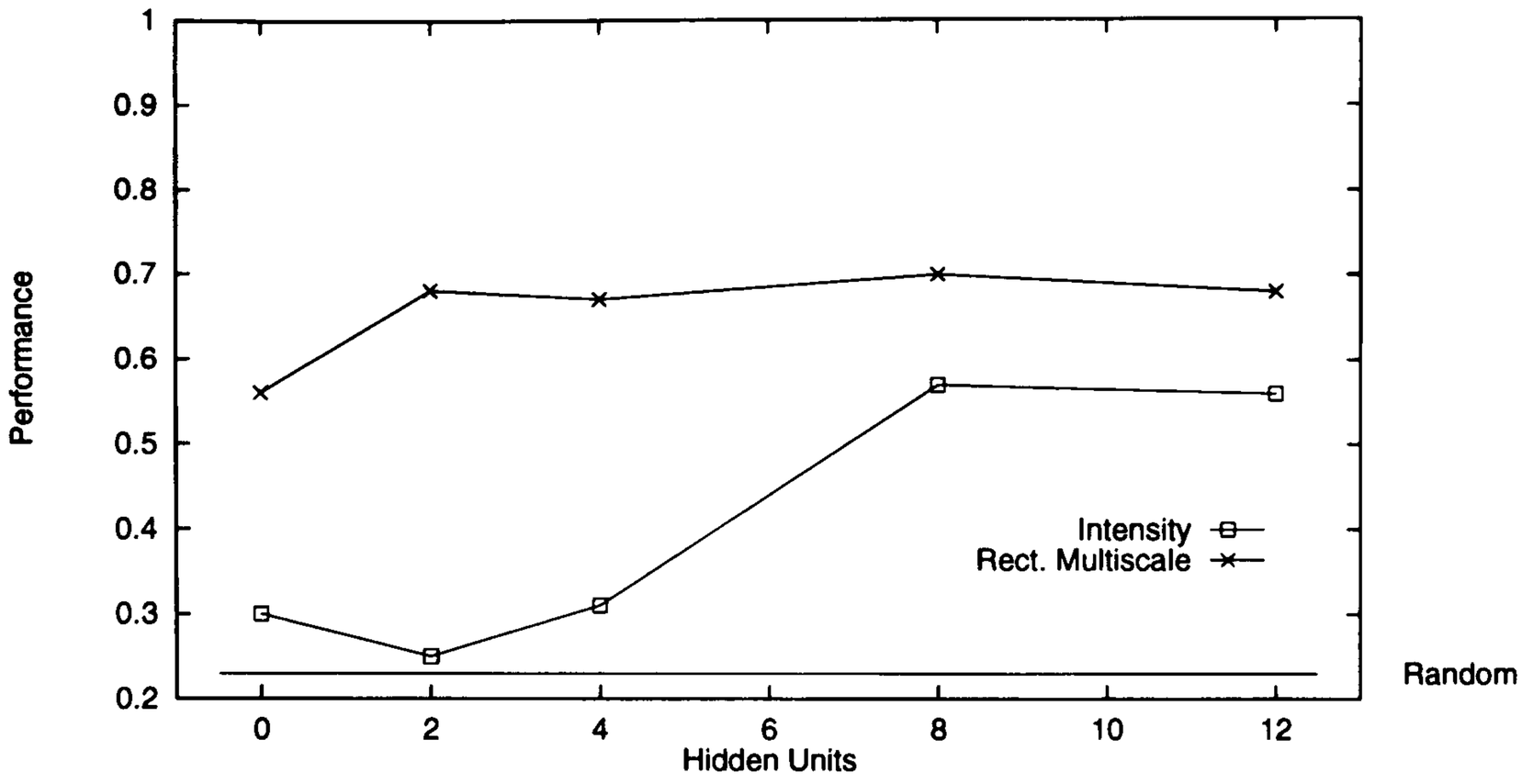


Figure 5.6: Performance as a function of network size for intensity, multiscale and rectified multiscale coding learning with independent sensory noise. Each data point is the mean performance of 3 animats. All standard errors ≤ 0.03.

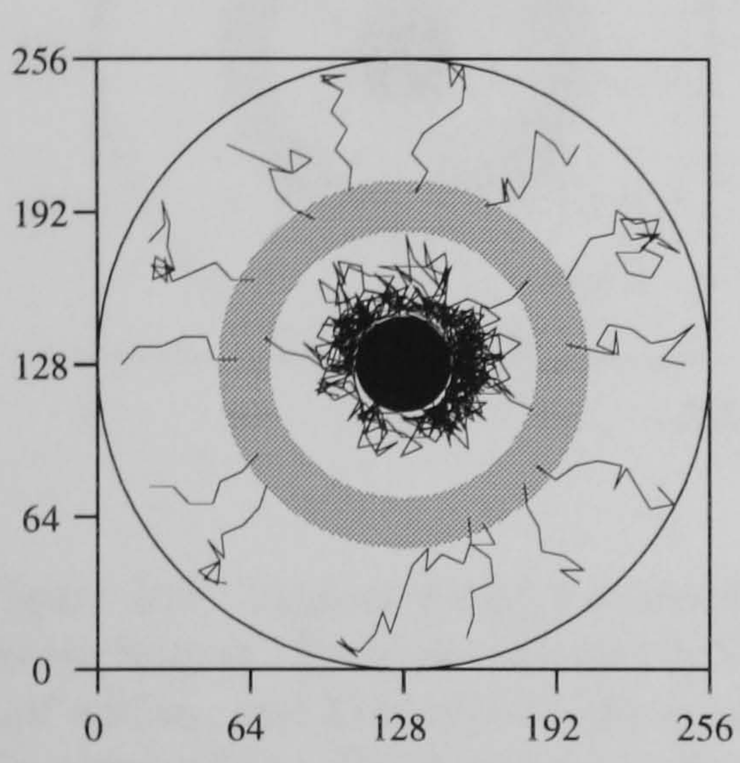
outward from within the goal ring. Thus, although intensity coding animats can easily learn to move directly toward the circle, they require 8 hidden units to learn the behaviour of moving away from the circle when the subtended angle is too large.

Rectified multiscale coding

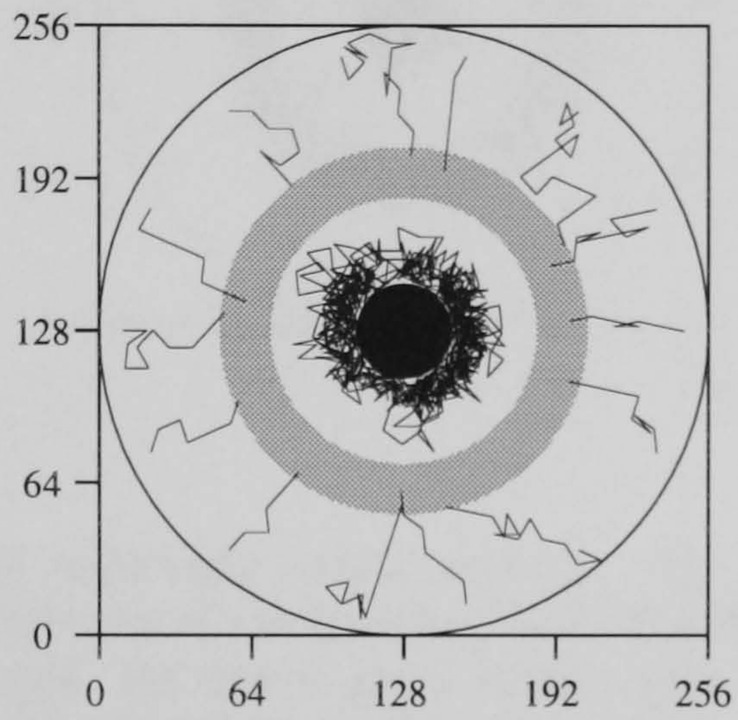
Fig. 5.8 shows typical paths for direct, and 2 hidden unit, rectified multiscale coding animats. Both animats move efficiently toward the goal region from all starting locations; however with two hidden units, the paths from within the goal ring are more direct. This is responsible for the significant differences in performance noted above. The behaviour of the 2 hidden unit animat is clearly more efficient at moving away from the circle when within the goal ring than that of either the direct rectified multiscale coding animat, or, any of the intensity coding animats of fig. 5.7.

Fig. 5.9a shows the behaviour of the 2 hidden unit animat when tested in a larger arena with a circle of radius 30, and without a goal region. The animats behaviour is unaffected by the switch to a larger arena, and it continues to search where the angle subtended by the circle is within the goal range of angles. Fig. 5.9b shows this animat's behaviour when tested with two circles of radius 20 and equal contrast. When the angles subtended by the circles are smaller than the goal range, animats move toward the nearest. When the angle subtended by one of the circles is larger than the goal range, the animats tends to be pulled toward the other, leading to more search at the correct distance from one of the circles, but in the direction facing the other circle.

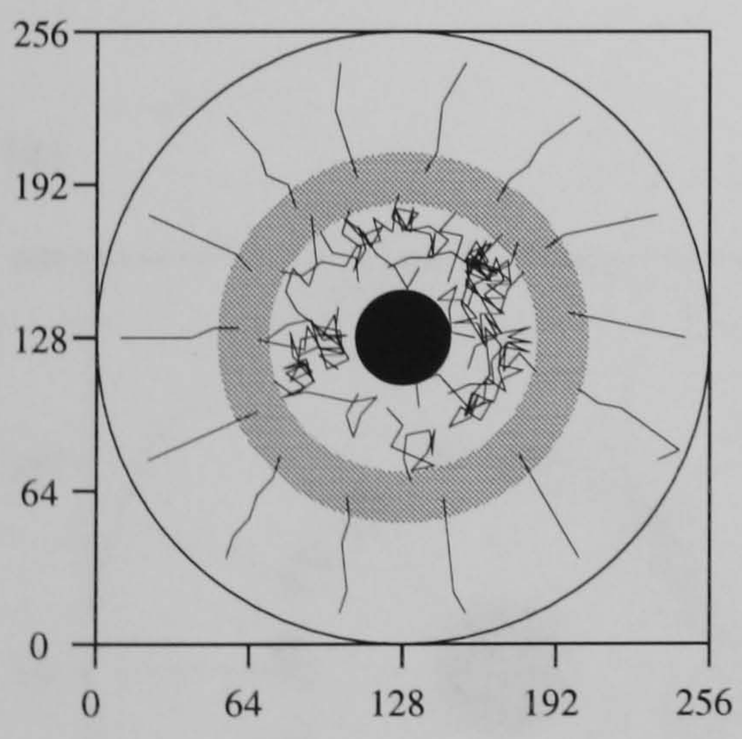
(a) Direct.
Performance = 0.31.



(b) 2 hidden units.
Performance = 0.44.



(c) 4 hidden units.
Performance = 0.60.



(d) 8 hidden units.
Performance = 0.75.

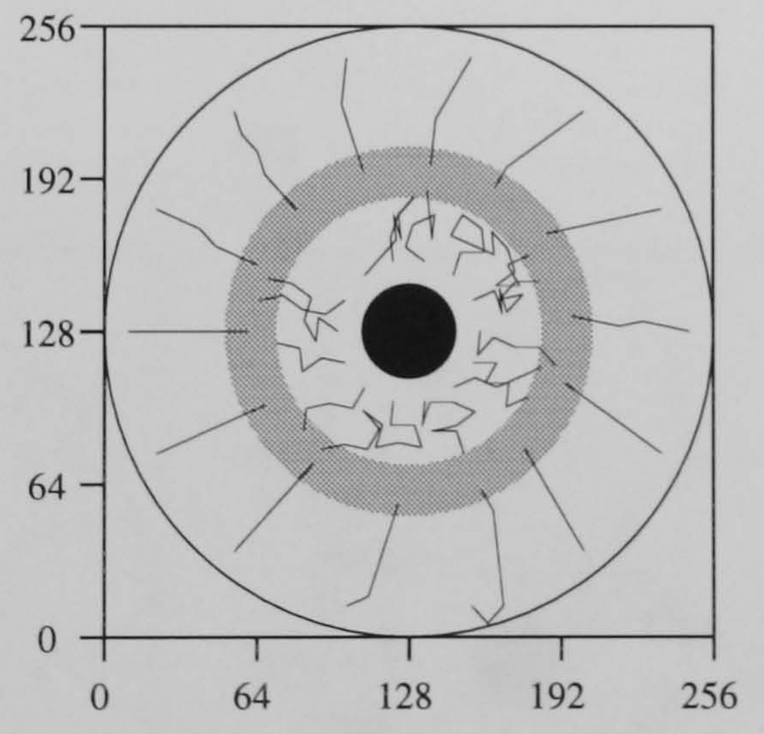


Figure 5.7: Typical paths for intensity coding animats. The invisible goal region is shown in gray. Each plot shows the behaviour of a single animat, from different starting locations: half within, and half outside the goal ring. Animats vary according to filter network size.
(For these plots : Circle radius = 20, circle intensity = 0.7, wall intensity = 0.2.)

(a) Direct.
Performance = 0.75.

(b) 2 hidden units.
Performance = 0.83.

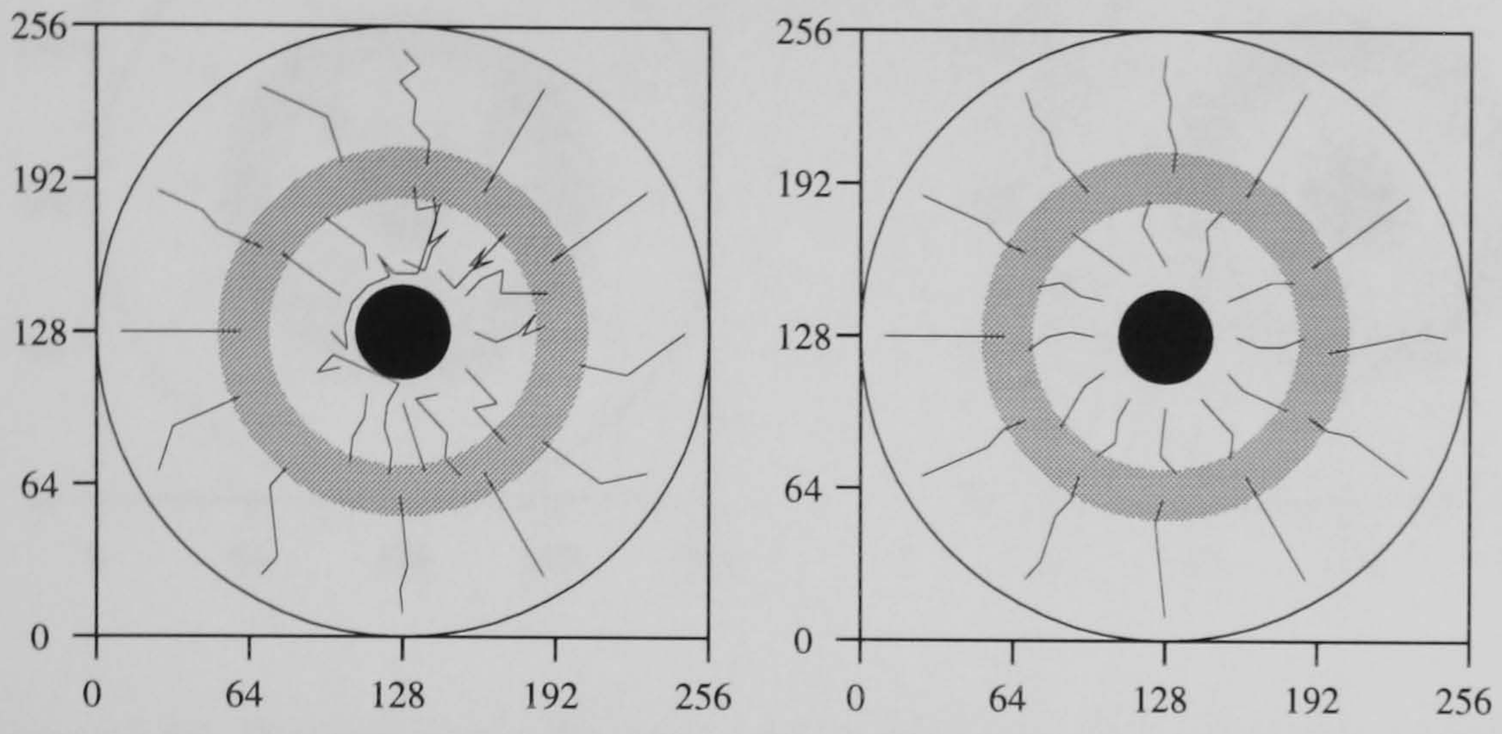


Figure 5.8: Typical paths for rectified-multiscale coding animats. The invisible goal region is shown in gray. Each plot shows the behaviour of a single animat, from different starting locations: half within, and half outside the goal ring. (a) Direct, (b) 2 hidden units.
For these plots : Circle radius = 20, circle intensity = 0.7, wall intensity = 0.2.

(a)

(b)

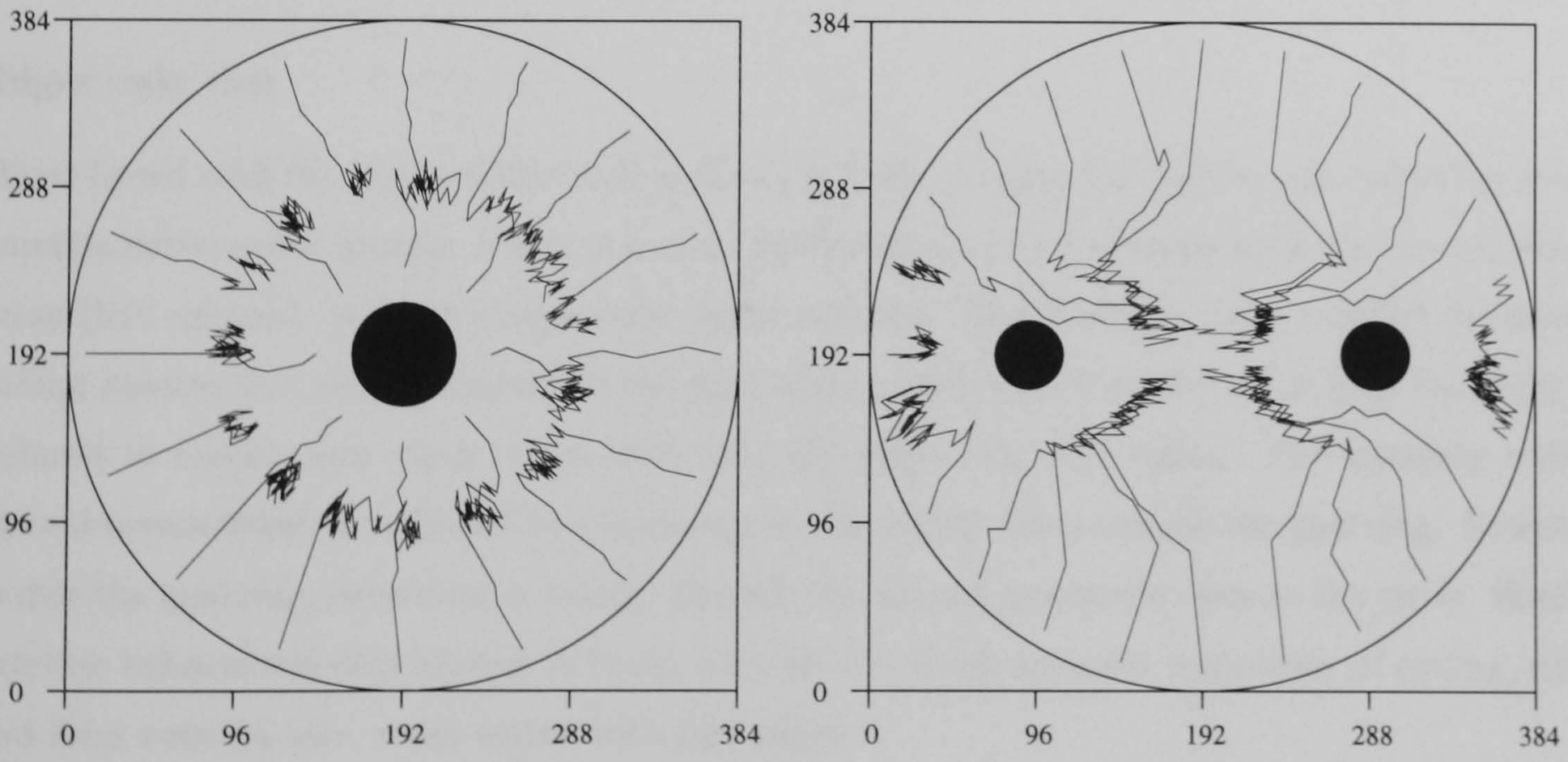
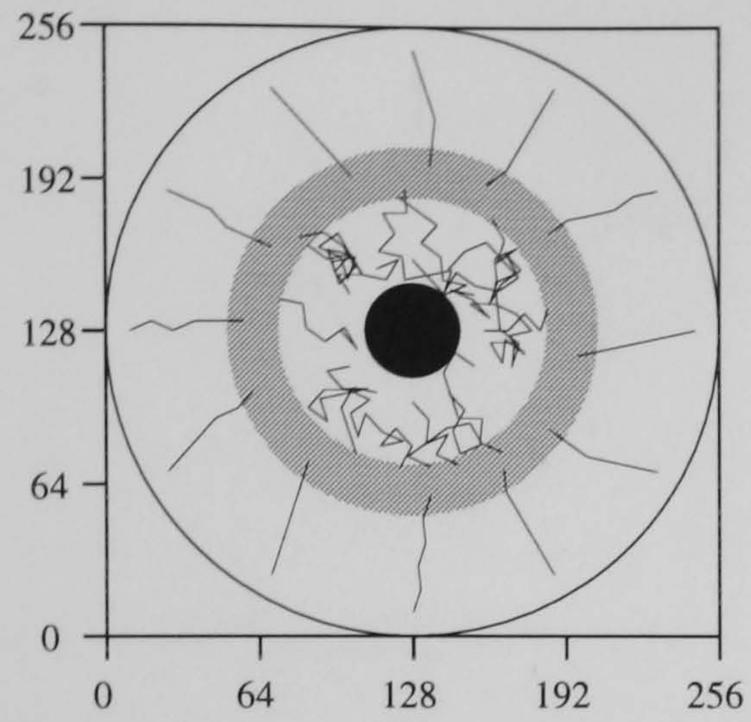


Figure 5.9: Paths of a 2 hidden unit, rectified multiscale coding animat tested in modified environments. (a) In larger arena than during learning (paths start at the arena edge and near the circle). (b) With two circles of equal contrast. Each path is 30 time steps long.

(a) 8 hidden unit, intensity coding animat. Performance = 0.62.



(b) 2 hidden, rectified multiscale coding animat. Performance = 0.69.

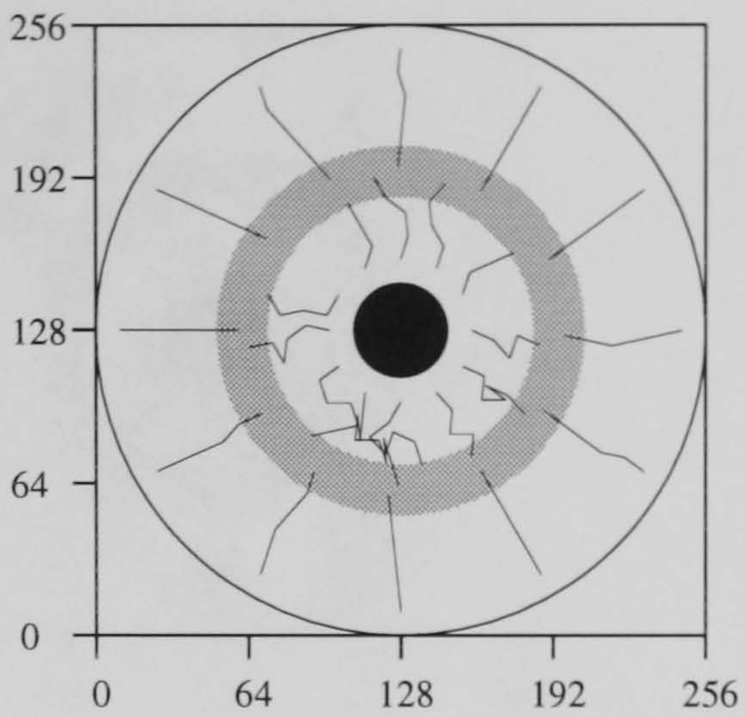


Figure 5.10: Typical paths for animats learning (and tested) with independent visual noise. Animats are from the best performing intensity (8 hidden units) and rectified multiscale (2 hidden units) coding conditions.
For these plots : Circle radius = 20, circle intensity = 0.7, wall intensity = 0.2.

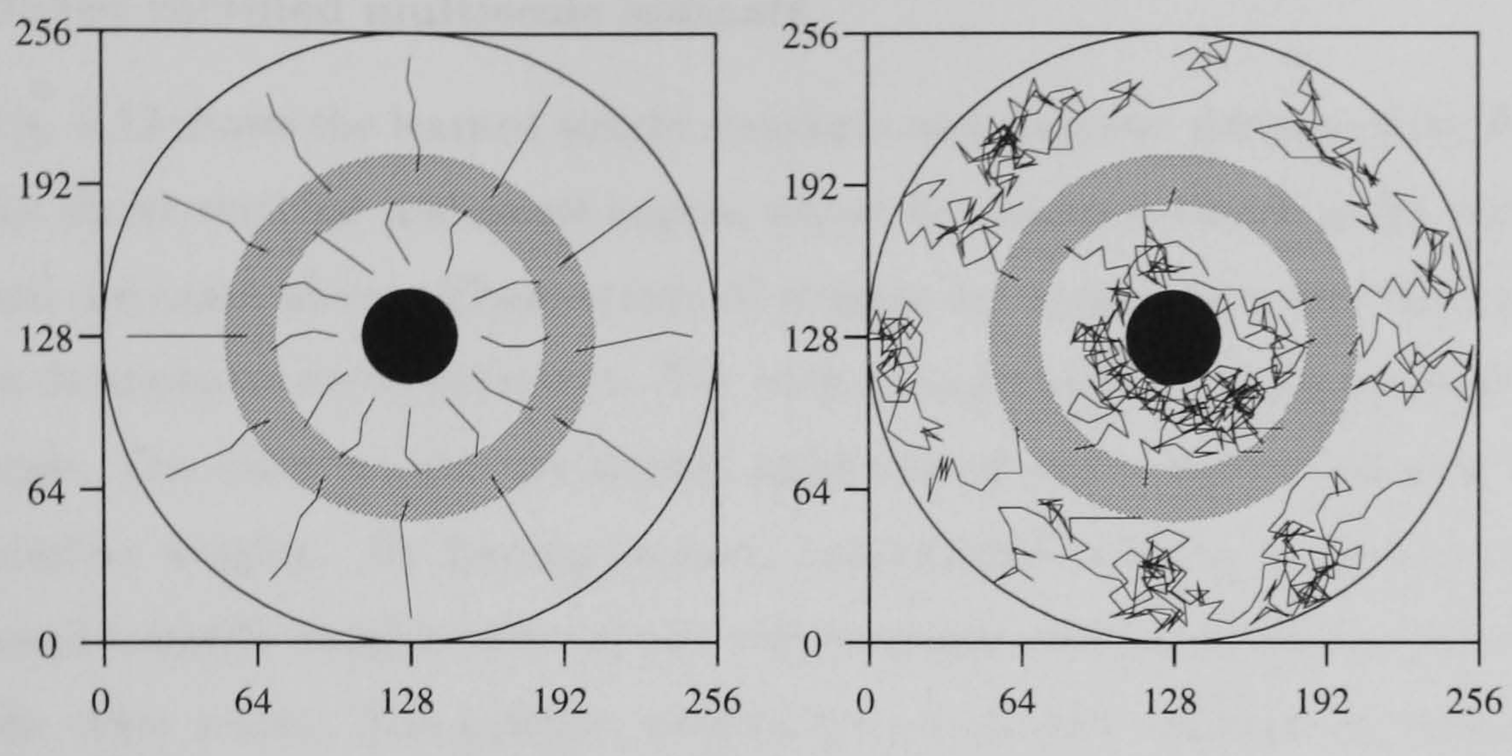
Independent sensory noise

Fig. 5.10 shows typical paths for 2 hidden unit, rectified multiscale coding, and 8 hidden unit, intensity coding animats that learned in the presence of independent sensory noise. As in the noiseless case, the rectified multiscale coding animat is able to move away from the circle, when required, more efficiently than the intensity coding animat despite having fewer hidden units.

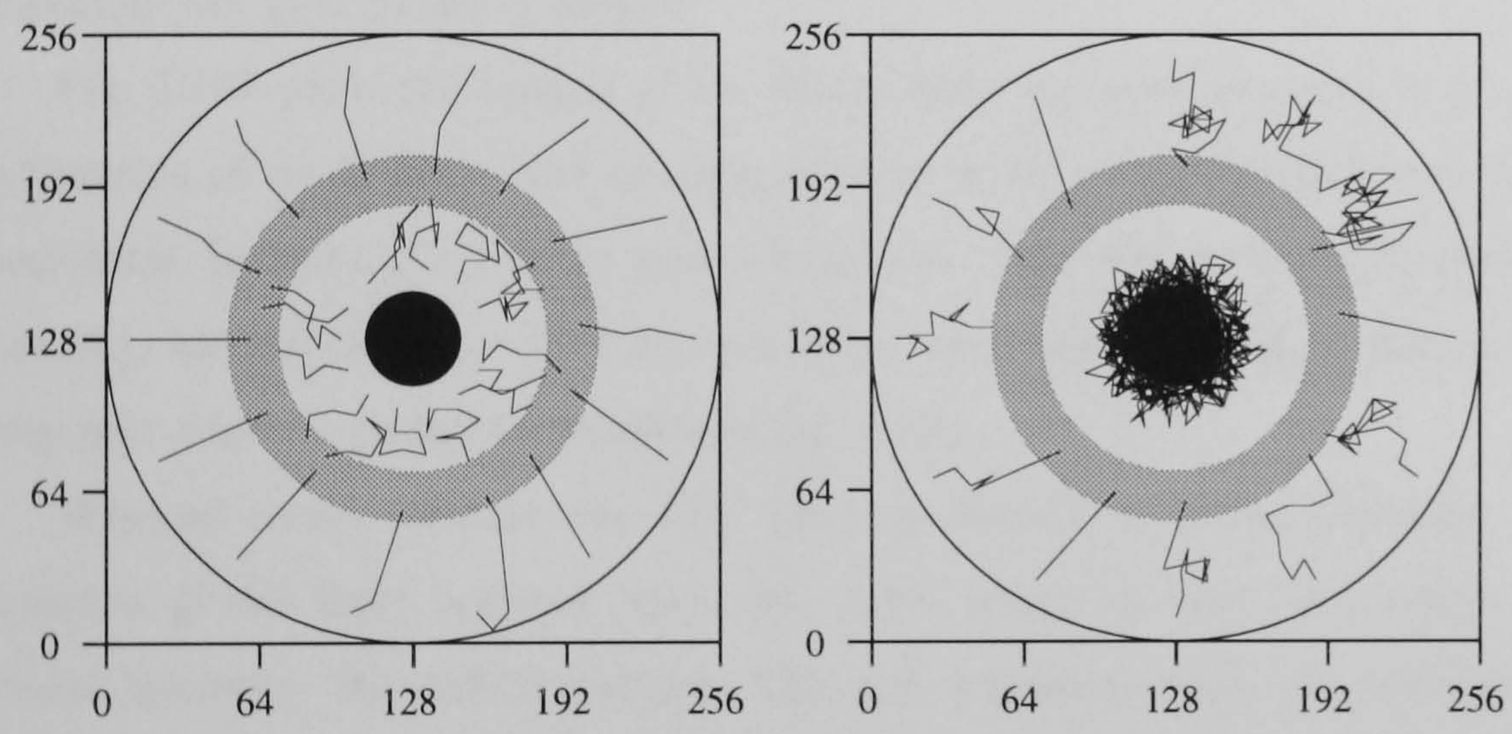
Edges only test

When tested with the image of the circle replaced by just its edges, the performance of all the above animats deteriorates greatly. Fig 5.11 shows typical behaviour of animats with the normal visual array (left column), and with edges-only (right column). The 2 hidden unit, rectified multiscale coding animat that moves directly to the goal when tested with the solid circle (fig. 5.11 left), is reduced to randomness when tested with only the edges (fig. 5.11 right). The intensity coding animat is somewhat less effected by the change to edges-only when outside the goal ring. However, within the goal ring, behaviour is totally altered; the animat just sticks close to the circle. Similar extreme behavioural degradation is found with all the above animats, regardless of coding, noise and filter network size, when tested with just edges.

(a) Zero visual noise. 2 hidden unit, rectified multiscale coding.



(b) Zero visual noise. 8 hidden unit, intensity coding.



(c) Independent visual noise, 2 hidden unit, rectified multiscale.

Figure 5.11: Typical paths of 3 animats in response to solid circle (left column), and edges-only circle (right column). Animats plotted here are from the best performing network size for each of the coding conditions. Behaviour of all animats is highly degraded by replacement of the solid angle with just its edges.

5.4.4 Internal structure

Direct rectified multiscale animats

Fig. 5.12 shows the learned weight structure and response pattern of the filter network controlling the direct rectified multiscale animat whose behaviour is shown in fig. 5.8a (performance = 0.75) and discussed above. These arrays of weights are convolved across the rectified multiscale arrays to determine output activation. The only strongly positive weights are on the third from coarsest scale. The other weights are mostly small valued and negative, but sum to slightly outweigh the positive weights. By having positive, concentrated weights on this scale, surrounded by many small negative weights, the output unit responds when activity on this scale exceeds activity on the other scales. The positive weights are all on scale number 3, corresponding to a LoG filter of spatial scale 13.3 degrees. As shown in fig. 5.2, scale 3 has the most activity when the circle subtends the goal range of angles.

Fig. 5.12b plots the output of the direct filter network in response to a circle, of radius 20, as a function of its distance and bearing relative to the filter. The left column plots the output unit activation between 0.0 (black) and 1.0 (white). The highest activity occurs in the region of zero bearing, around distance 50–120, and fades away with increasing distance. This is the expected response pattern given the weights of fig. 5.12a.

Without noise, because the LoG filter is insensitive to the absolute visual intensity, the activation of the filter network when the circle is not in view, is a single number irrespective of visual intensity; the null activation. The null activation links the network output to the animats behaviour. When the output activity in response to the circle is higher than the null activation, the animat stochastically moves toward the circle. When the output activity is lower than the null activation, the animat stochastically moves away from the circle. Hence, the null activation enables output activation plots, such as the left column of fig. 5.12b, to be understood in behavioural terms. The right column shows the output activation, thresholded so that all those below the null activation are set to zero. Hence, it is a plot of the distances and bearings of the circle with respect to a filter network which lead to the animat moving in the direction of that filter network. For a circle of radius 20, the distance range corresponding to the goal subtended angle range is 55–77. It can be seen that this filter network does not respond when the circle is closer than about 60, and so the animat is unlikely to move toward a circle closer than this. Response, and hence probability of moving toward the circle, peaks at around distance 80, fading away with increasing distance.

The relationship between network output and behaviour is made clearer in fig. 5.12c, which plots the response of the network to a centered circle, as a function of distance. When distance is large, and hence subtended angle small, output activation is greater than the null response and so the animat moves toward the circle. As distance decreases, activation gradually increases,

reflecting the decreasing expected number of steps to the goal. When subtended angle exceeds the goal region, output is lower than the null response, and so the animat is most likely to move away from the circle. Notice that in the region just closer to the circle than the goal region, output is still higher than the null response and so the animat will tend to move toward instead of away from the circle. Approximately the same learned weight structure and response profile were shown by all direct animats using rectified multiscale coding.

Individual filter networks have overlapping receptive fields within the convolution architecture that determines animat behaviour. In order to see how individual response fits in with the rest, fig. 5.13 plots the motor array activity (thresholded at the null activation) in the direction nearest to the circle placed at all distance and directions from the animat. Although most of the space is covered, there are clearly holes, especially between the distances of 60 and 100, and at far distance.

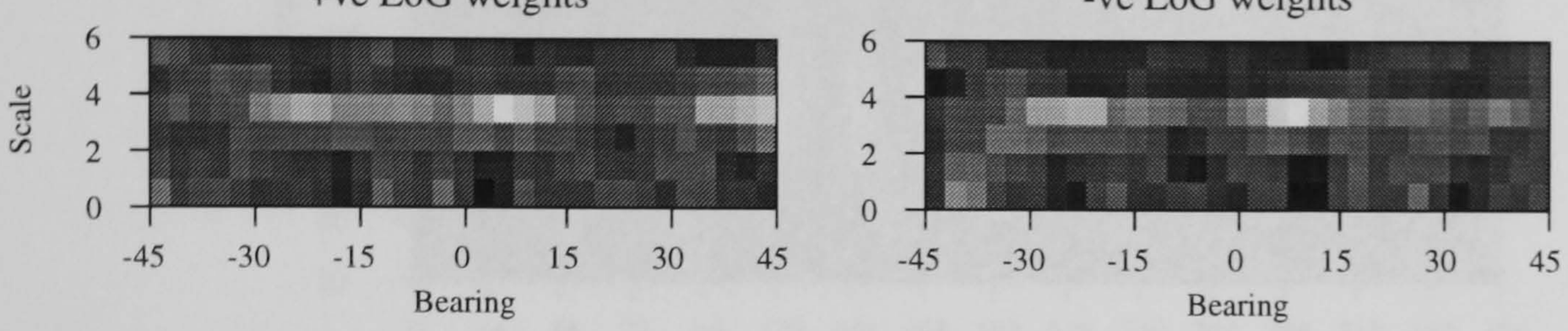
2 hidden unit rectified multiscale animats

Fig. 5.14 shows the learned weights and response profile for the 2 hidden unit, rectified multiscale animat, whose highly efficient behaviour was shown in fig. 5.8b. The behaviour leads to performance significantly better than that of the direct, rectified multiscale animats discussed above.

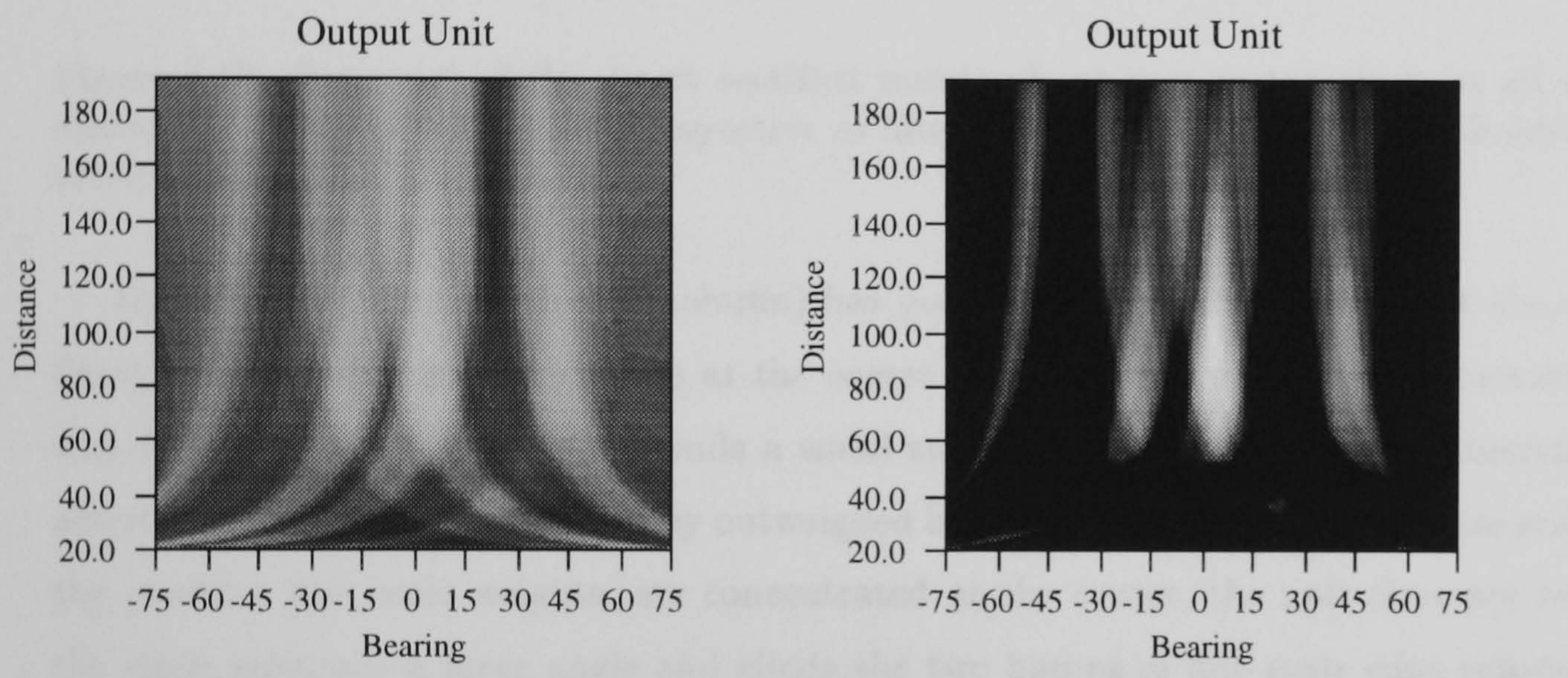
Fig. 5.14a and b, show the weight structure learned by the filter network. Each hidden unit has 2, two-dimensional arrays of weights; for the positive and negative components of the multiscale filtered visual array. For both hidden units, weights for the positive and negative parts of the filtered image are very similar. This pattern is seen in most rectified multiscale animats in both this and the last chapter, and is due to the symmetry about zero of the filtered sensory array, given the visual conditions imposed by the solitary circle environment. The circle intensity is equally likely to be higher or lower than the wall intensity, and so, for a unit to respond to a feature, such as subtended angle, irrespective of contrast, the same weight structure must be learned for both the negative and positive filtered arrays.

Hidden unit 1 (fig. 5.14a, left column), has positive weights concentrated on the 2 coarsest scales, and negative weights concentrated around the center of the finer scales. The result of this structure is that when the circle is far away, and hence causing activity at the finest scales, this hidden unit is suppressed. When the circle is nearby, and hence subtends a large angle, the activity on the coarse scales outweighs the activity at the the fine scales and so the unit becomes active. This pattern is shown in fig. 5.14a (right column), which plots the activity of the unit in response to a circle, radius 20. Around the center of the unit's receptive field, the unit only becomes active when the circle's distance is less than about 60. The unit is also active when the bearing of the circle is more than 30 degrees; diminishing with distance. This is due to the more extended pattern of activity at the coarse scales compared with the fine scales.

a) Filter network weights. Output unit bias = 0.62.



b) The response of the output unit to a circle, radius 20.



c) Output unit response at zero bearing.

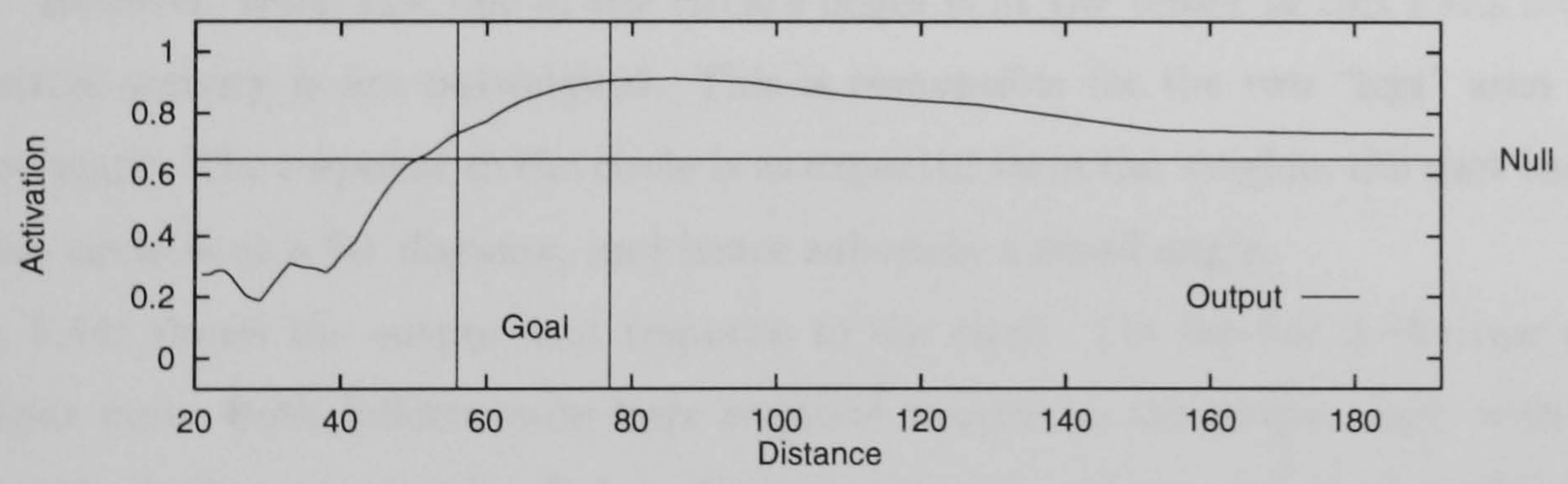


Figure 5.12: Learned weight structure and response pattern of a direct rectified multiscale coding filter network (animat performance = 0.75). a) The direct filter network consists of 2, two-dimensional arrays of weights to the single output unit. b) The activation of the output unit in response to the circle, of radius 20, as a function of its distance and bearing relative to the filter. The left column shows the unit activation, between 0 (black) and 1 (white). The right column displays this in more behaviourally relevant terms by thresholding at the null activation level. c) The response of the output unit to the centered circle as a function of distance (ie. the central column of left image in b)). The vertical lines mark where the angle subtended by the circle is within goal range. Null marks the output of the filter network when the circle is out of view. Where the output is greater than null, the animat moves toward the circle; where the output is less than null, the animat moves away from the circle.
(For these plots: Circle radius = 20, circle intensity = 0.7, wall intensity = 0.2.)

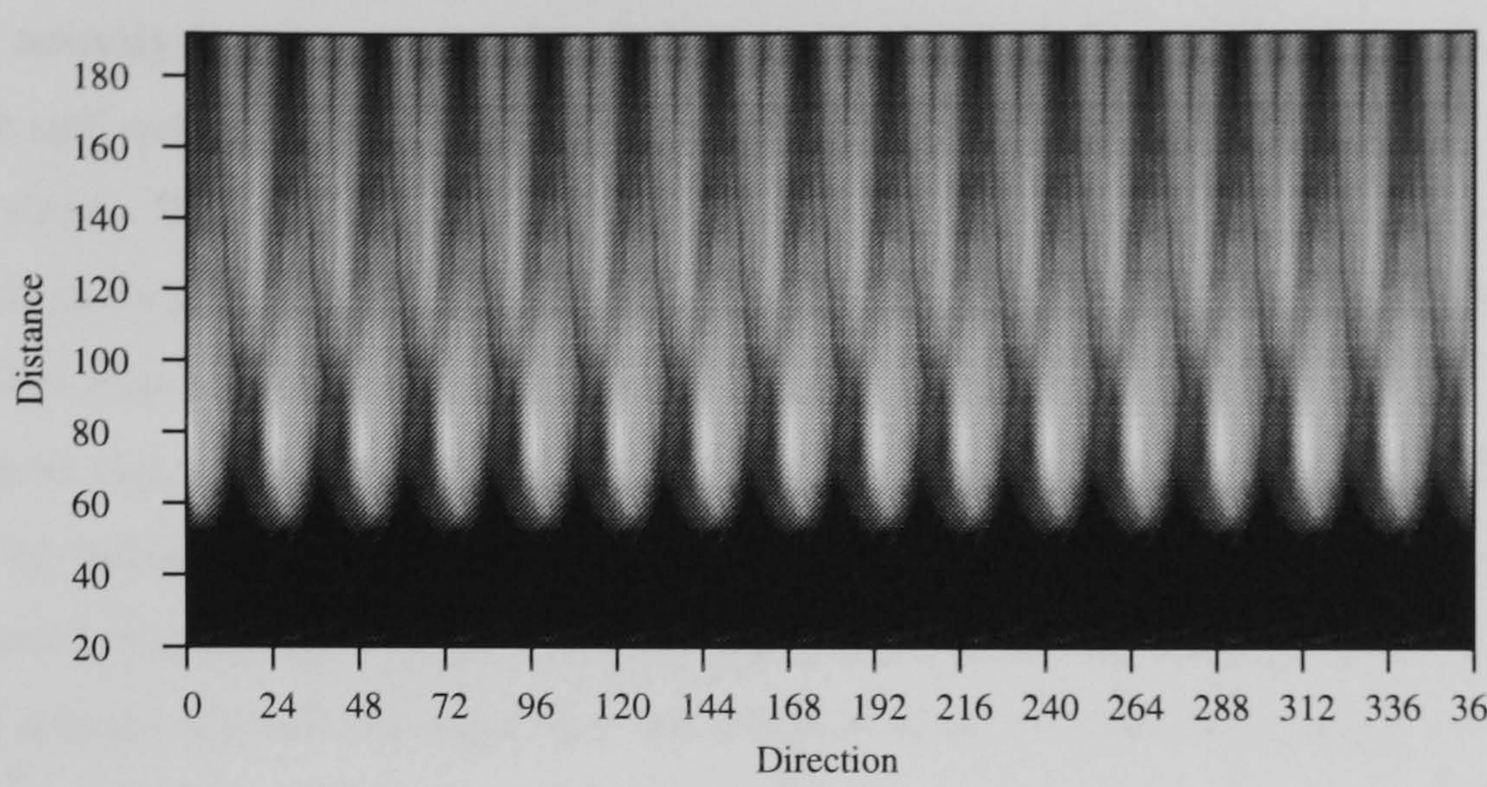


Figure 5.13: Response of the direct rectified multiscale animat to the circle at all distance and directions. Response is plotted irrespective of filter network direction and thresholded at the null activation.

Hidden unit 2 (fig. 5.14b, left column) has positive weights concentrated at the center of the finest scales, with negative weights at the coarse scales. The result of this structure is that the unit is active when the circle subtends a small angle. As the subtended angle increases, positive activity at the fine scales becomes by outweighed by negative activity at the coarse scales. Because the positive, fine scale weights, are concentrated at the center, the unit does not respond when the circle subtends a large angle and elicits the two humps of fine scale edge response shown in fig. 5.2. However, when just one of the circle's edges is at the center of this units receptive field, the positive activity is not outweighed. This is responsible for the two "legs" seen in fig. 5.14b (right column). The response to the circle is as expected from the weights; the unit becomes active when the circle is at a far distance, and hence subtends a small angle.

Fig. 5.14c shows the output unit response to the circle. On the left is the raw activation of the output unit. Both hidden units have negative weights to the output unit, with hidden unit 1 having the higher magnitude. Below distance 120, the output activation profile is essentially the negative of hidden unit 1's profile. At distances of greater than 120, hidden unit 2 inhibits the output unit activity by an amount increasing with distance. A more behaviourally relevant rendering of the output activity is shown in the left column, where it is thresholded at the null activation level (the response of the filter network to a flat visual input), to indicate where the animat is more likely to move toward the circle. The region of high response is restricted to within 30 degrees of the center of the filter networks receptive field, with activity rising sharply at around distance 60, and then decreasing slowly with increasing distance. Comparing this plot with the corresponding one for the direct rectified multiscale animat (fig.5.12b), it is clear that with 2 hidden units a more compact and accurate region of high activity is learned.

The relationship between stimulus, hidden unit, and output activity is made clearer in fig. 5.15,

which plots activity in response to the circle centered within the networks receptive field. Also shown is the null activity level. Where the output is greater than null, the animat tends to move toward the circle. Where the output is lower than null, the animat tends to move away. At far distance, output is higher than null and increases as the distance to the goal decreases. Output activity in this region is determined by hidden unit 2. Output drops to well below the null level within the goal region, and remains very low at short distance. Output activity in this region is determined by hidden unit 1.

This learned weight structure is interesting because the network has learned to partition the response to sensory stimuli amongst the two hidden units, so that one responds to the circle at a far distance, and the other at near distance. The output unit can then combine the hidden unit responses to produce the output leading to most effective behaviour. The other 2 animats simulated in this condition learned a similar stimuli decomposition. This computational strategy is not available to direct filter networks; hence their significantly poorer performance. That this learned stimuli partition is sufficient for the task is indicated by animats with more hidden units doing no better than those with 2 hidden units.

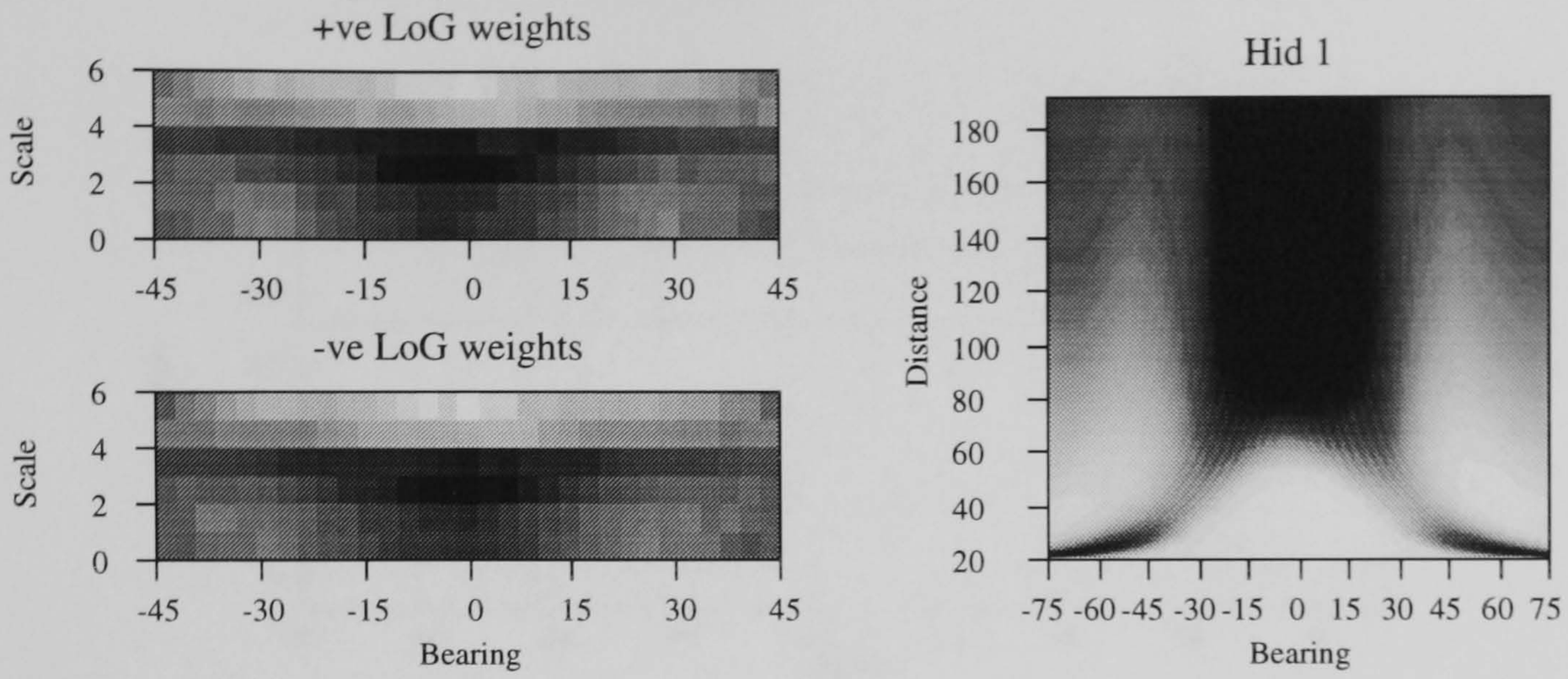
Fig. 5.16 shows the combined response of this animat to the circle at all distance and directions. The response of individual filter networks overlap to provide total coverage of the space. Comparing this with the same plot for the direct rectified multiscale animat (fig. 5.16) indicates why this animat performs significantly better.

5.4.5 Response to edges-only circle

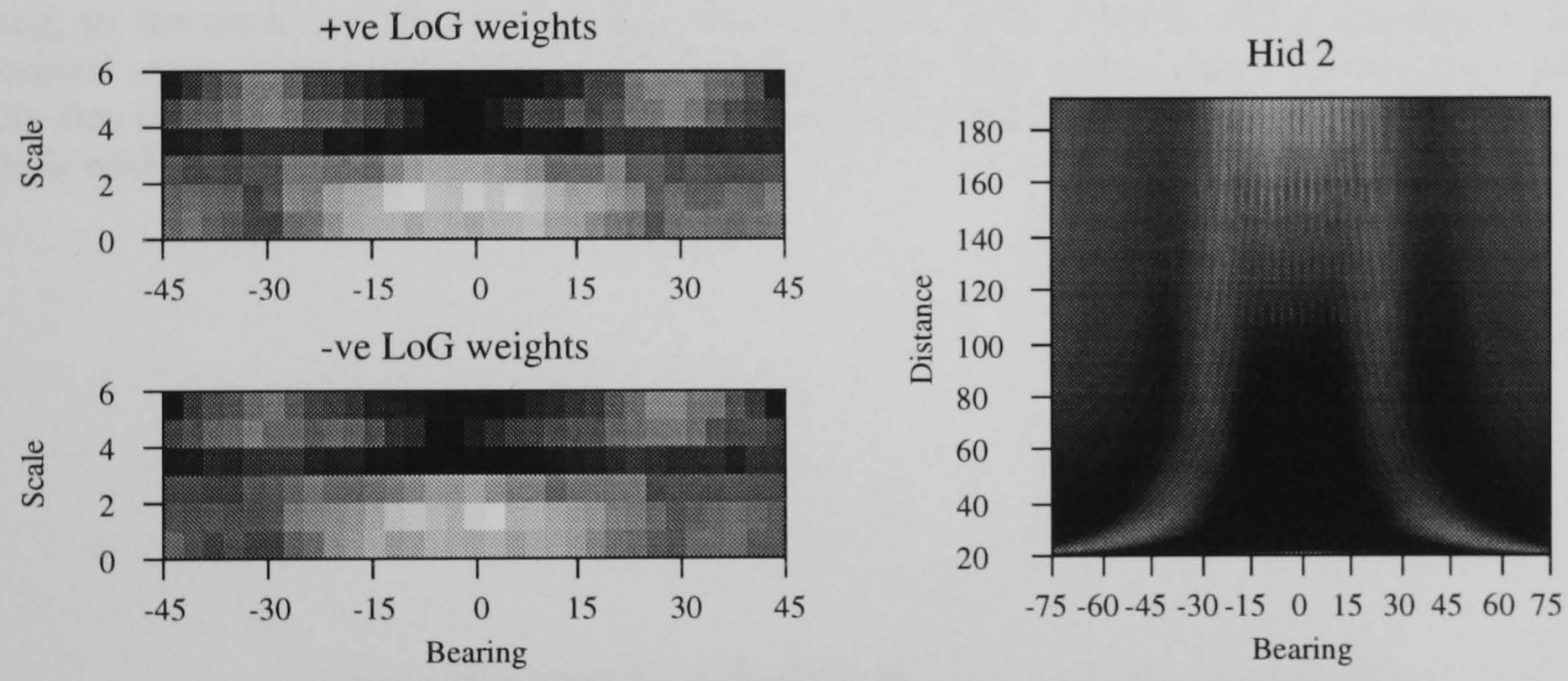
In section 5.4.3 it was shown that when the solid circle image is replaced with the edges-only circle, the performance of all animats, unlike honeybees, is extremely degraded. Now, having understood what the rectified multiscale animats have learned to compute that enables them to perform so efficiently, it is possible to understand why they are so degraded by the edges-only test.

As explained in section 5.4.4, the direct rectified multiscale animats learn to concentrate positive weights on a single, relatively coarse scale, with other weights small and negative. The positive weighted scale roughly matches the angle subtended by the circle from within the goal ring. When the solid circle image is replaced by just its edges, maximal activity is only elicited at the fine scales, regardless of distance. Hence, the direct filter network's coarse scale strategy is strongly affected by replacement of the solid circle with its edges. Fig. 5.17 shows the response of the network filter to the edges only image, and this is clearly very different to the response to the solid circle shown in fig. 5.12. Hence the massive performance degradation shown by the animats in the edge only test. Only at far distance are the response patterns similar; this is because the difference between the solid and edge only images is small at far distance, and both images will match when the circle subtends 2, or less, receptors.

a) Hidden unit 1: Weights (bias = -2.44, weight to output = -4.50) and response to a circle.



b) Hidden unit 2: Weights (bias = 0.68, weight to output = -2.58) and response to a circle.



c) The response of the output unit to a circle: Raw and thresholded at null activation

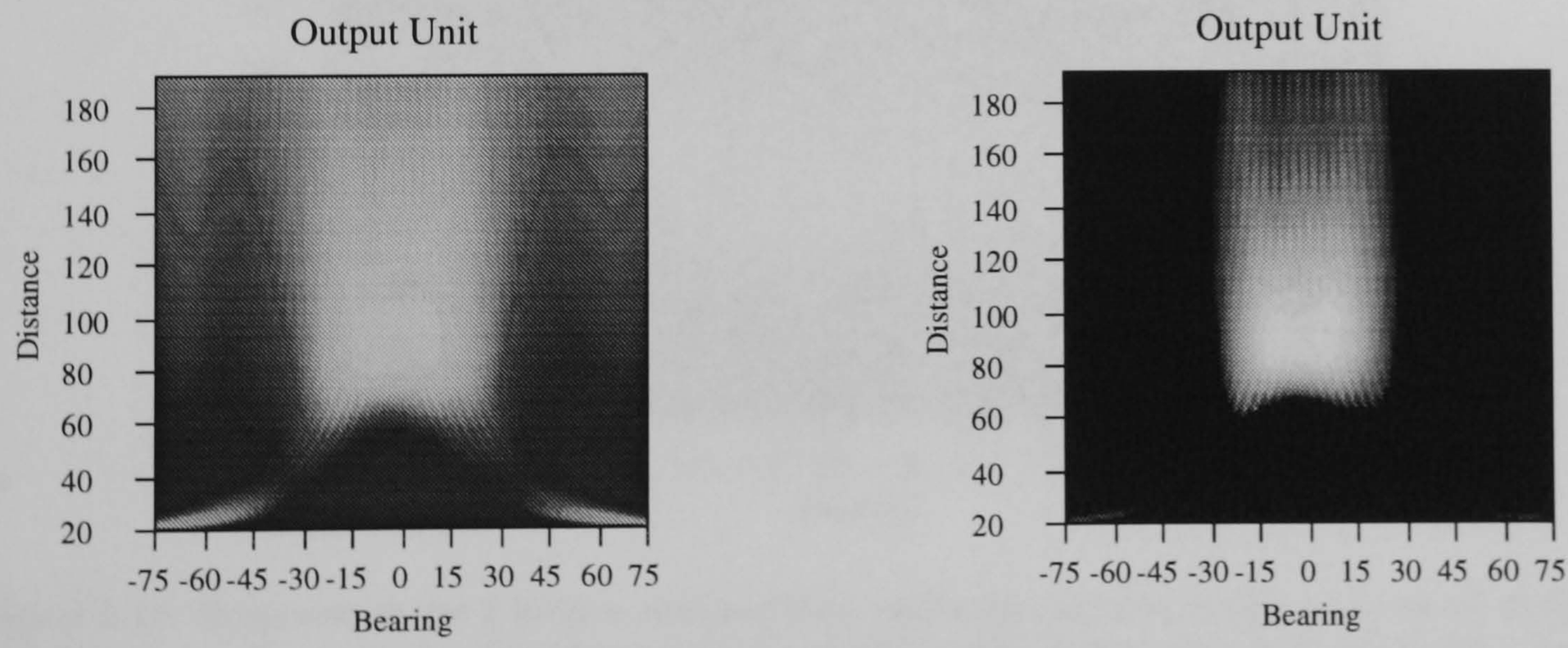


Figure 5.14: Learned weight structure and response pattern of a 2 hidden unit rectified multiscale coding filter network (animat performance = 0.85). a) and b) Each hidden unit has 2, two-dimensional arrays of weights one for the positive and one for the negative multiscale arrays (left column). The right column plots the activation of the hidden unit in response to a circle, radius 20. c) The output activation in response to the circle (left column), and thresholded at the null activation level (right column).
(For these plots: Circle radius = 20, circle intensity = 0.7, wall intensity = 0.2.)

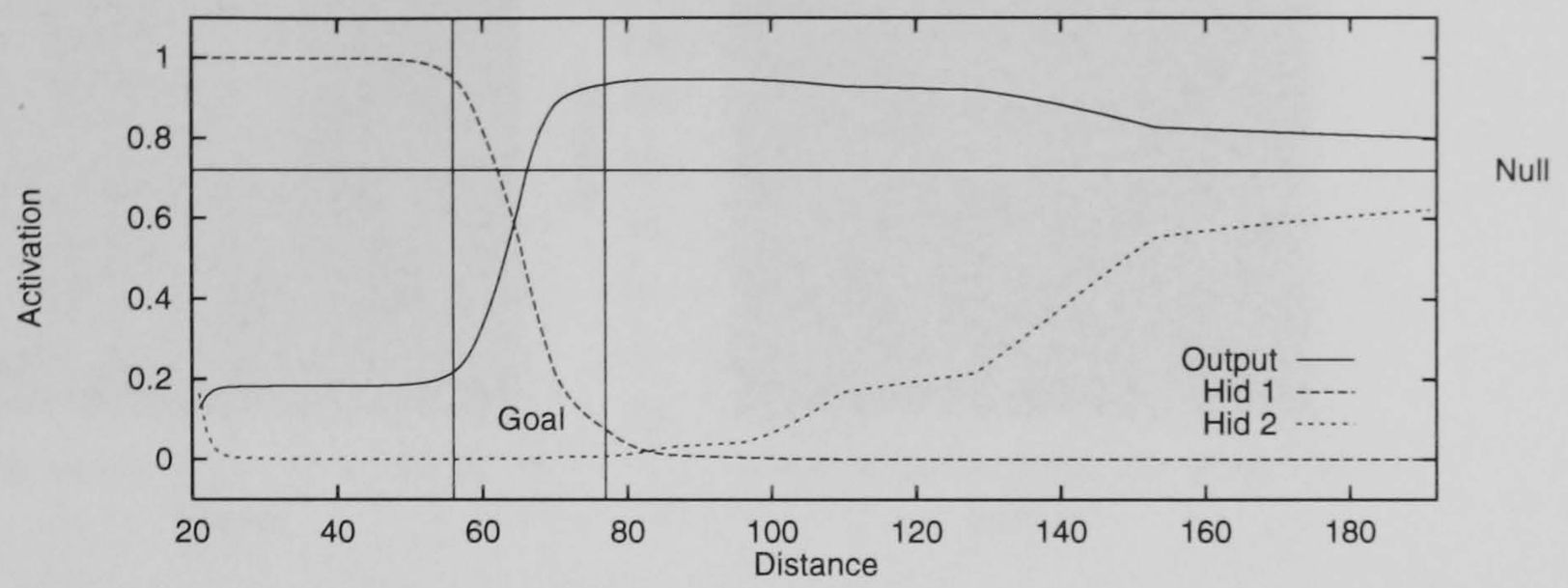


Figure 5.15: The activation of the output unit of a 2 hidden unit animat using rectified-multiscale coding, to the circle placed at the center of its receptive field. Hidden unit 1 responds to large subtended angle, hidden unit 2 to small subtended angle. The output unit combines these into a utility function, suitable for the task. The vertical lines mark where the angle subtended by the circle is within goal range.

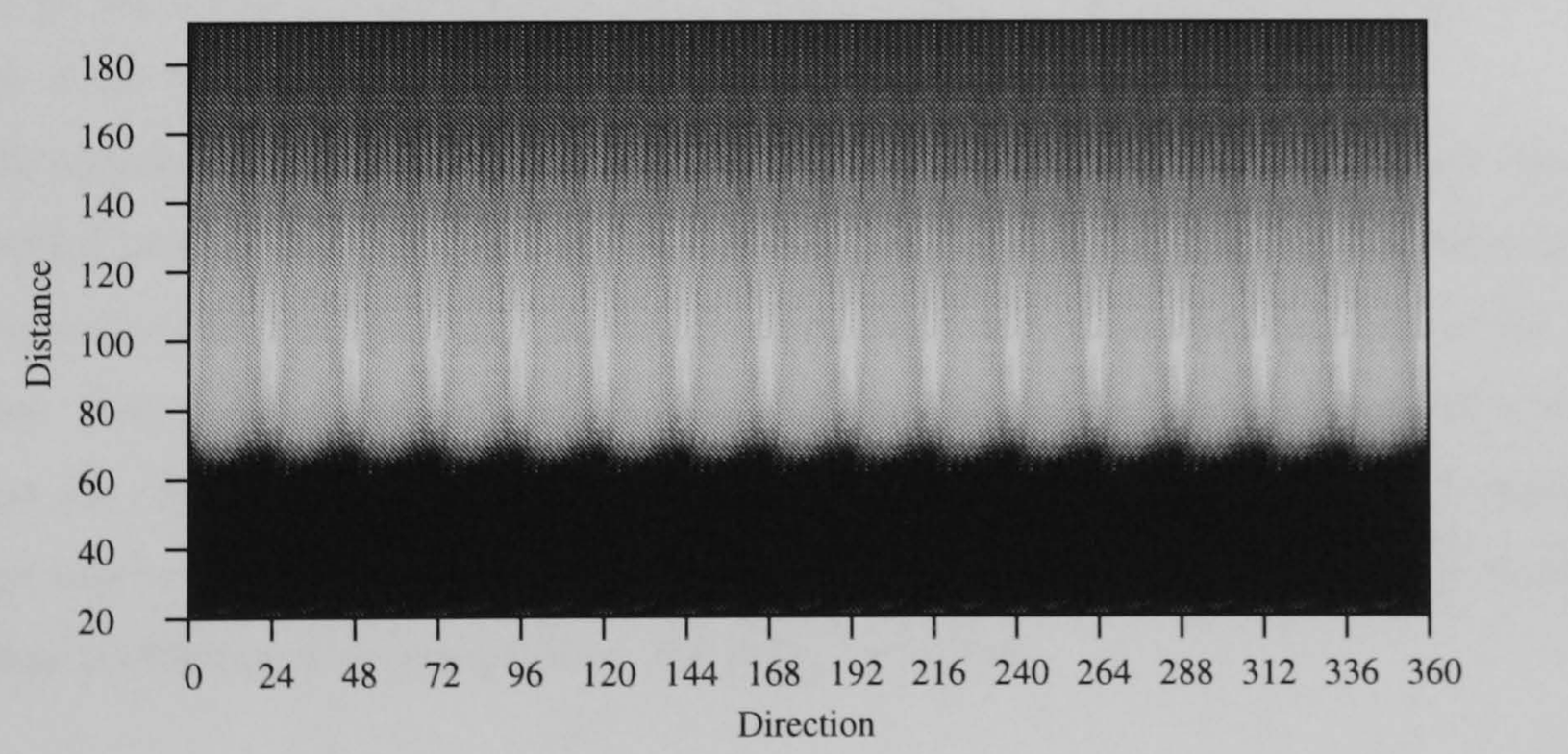


Figure 5.16: Response of the 2 hidden unit rectified multiscale animat to the circle at all distance and directions. Response is plotted irrespective of filter network direction and thresholded at the null activation.

Output unit response, raw and thresholded at null activation, to edges-only circle.

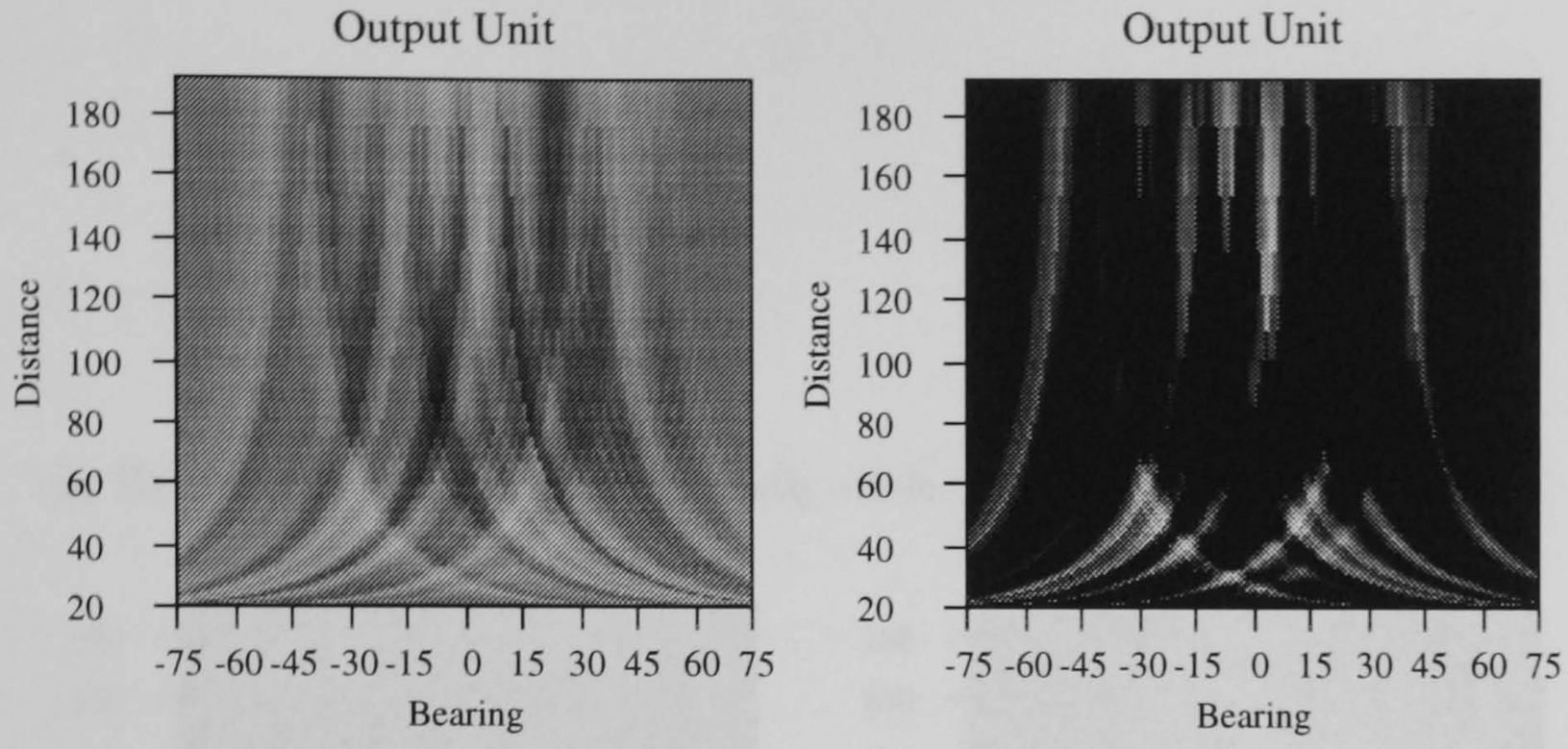


Figure 5.17: Response of the direct, rectified multiscale coding filter network to edges-only circle. This filter network is the same as in the earlier figure and discussed in the text. The large difference in response pattern, compared with the earlier figure, is due to the stimuli difference.

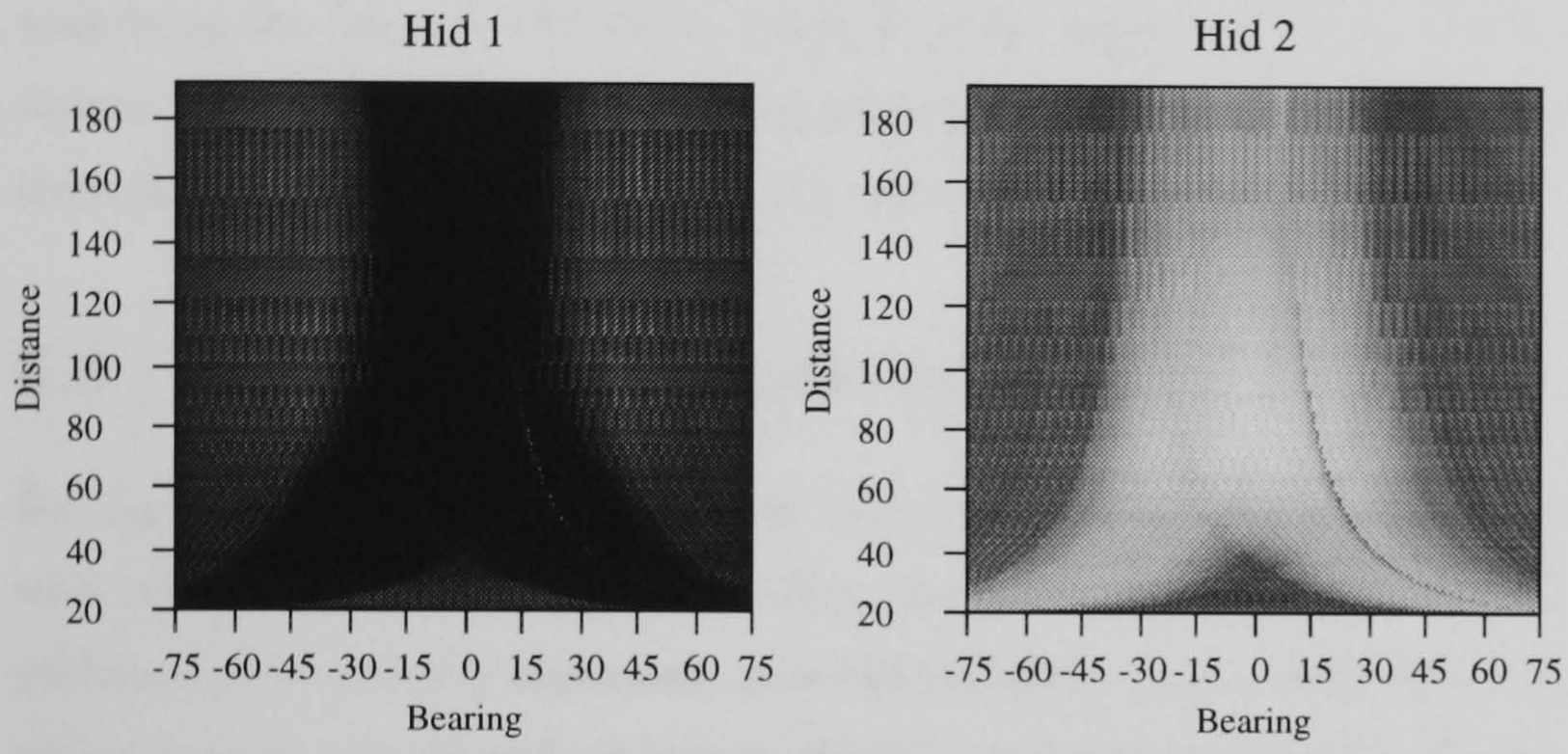
With 2 hidden units, the situation is similar. Fig. 5.18 shows the response pattern of the 2 hidden unit filter network of fig. 5.14 to the edges-only circle, and again there is a large difference. As explained in section 5.4.4, hidden unit 1 has learned to respond to the coarse scale activity elicited when the solid circle is nearby. Hence, as edges only elicit fine scale activity, it does not get active at all in response to the edges only circle. Hidden unit 2 has the opposite problem; it responds when activity at the fine scale outweighs activity at the coarse. Hence, it responds to the edges only circle regardless of distance. The output unit activity only exceeds the null level when the circle subtends a very small or a very large angle — totally inappropriate for efficient performance of the task.

The performance degradation shown when the solid circle is replaced by just its edges occurs because rectified multiscale animats consistently use activity at the coarse spatial scales as part of their learned computational strategy. The edges-only circle elicits strong activity at only the fine spatial scales. Hence the behavioural sensitivity of the animats to the replacement of the solid circle by just its edges. Rectified multiscale animats that learn in the presence of independent sensory noise rely at least as much on the coarse scales, and so the same computational argument underlies their performance degradation in the edges only test.

5.4.6 Conclusion

Without noise, 2 hidden unit, rectified multiscale animats perform significantly better than direct rectified multiscale animats, and this performance is not improved upon with more hidden units. Their performance of 0.82 is significantly better than intensity and multiscale coding animats, regardless of network size. Examination of behaviour shows that these animats can efficiently move

(a) Hidden unit response to edges only circle.



(b) Output unit response, raw and thresholded at null activation, to edges only circle.

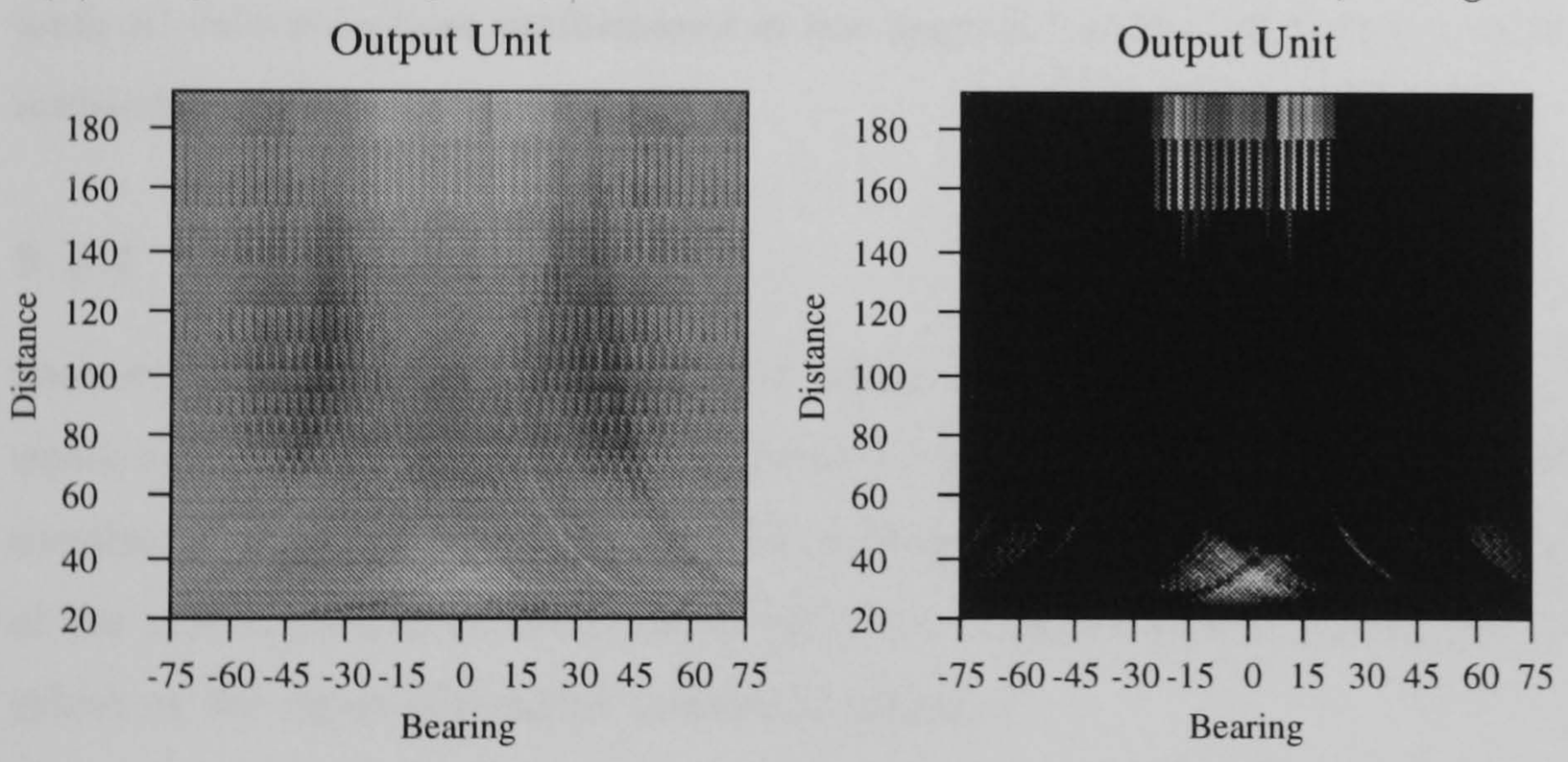


Figure 5.18: Response of the 2 hidden unit, rectified multiscale coding filter network to edges only circle. This filter network is the same as in the earlier figure and discussed in the text. The large difference in response pattern, compared with the earlier figure, is due to the stimuli difference. (a) Hidden unit activation. (b) Output unit activation.

to the goal ring from both inside and outside it. With independent noise, a similar situation holds with 2 hidden unit, rectified multiscale animats again outperforming direct, rectified multiscale and all intensity coding animats.

The computations learned by the rectified multiscale animats, that result in efficient performance with the solid circle, are highly sensitive to replacement of the solid region with only its edges. This is different from the behaviour shown by bees in Cartwright and Collett's (1983) experiment, and so all the above animats must be rejected as possible models of the computations underlying the insect's behaviour. Analysis of the learned internal structure of the animats in section 5.4.4 reveals the computational mechanisms underlying their behaviour, and explains the large deterioration of performance in the edges-only condition.

5.5 Rectified single scale coding

Having established above that rectified multiscale coding animats significantly outperform those with either intensity, or multiscale coding, the next step is to determine whether their superior performance is critically dependent upon the multiscale aspect of the filtering, or whether filtering with a LoG at a single scale is just as effective. A further question is whether single scale coding leads to animats whose performance is less degraded in the edges-only condition than those with mulitscale coding.

5.5.1 Method

Animats were simulated with rectified single scale LoG filters, and zero visual noise. 6 single scales numbered (0–5), each corresponding to one of the scales in the rectified multiscale case were simulated. Rectification of the single scale filtered array yields 2, 1-dimensional arrays, each a row of the corresponding rectified multiscale array. Thus, rectified single scale coding animats have subset of the input of rectified multiscale animats.

5.5.2 Performance

Fig. 5.19 shows the performance after 50k learning trials for less than 8 hidden units, and 100k learning trials for 8 hidden units, by which time all had converged. Also shown in fig. 5.19, for comparison, is the mean performance of direct, and 2 hidden unit, rectified multiscale coding animats.

Scale 0 corresponds to the finest of the rectified multiscales, and scale 5 to the coarsest. With the coarsest filter (scale 5), performance is very low regardless of network size. At the next from coarsest scale (4), performance is low for direct or 2 hidden unit filter networks, but increases to

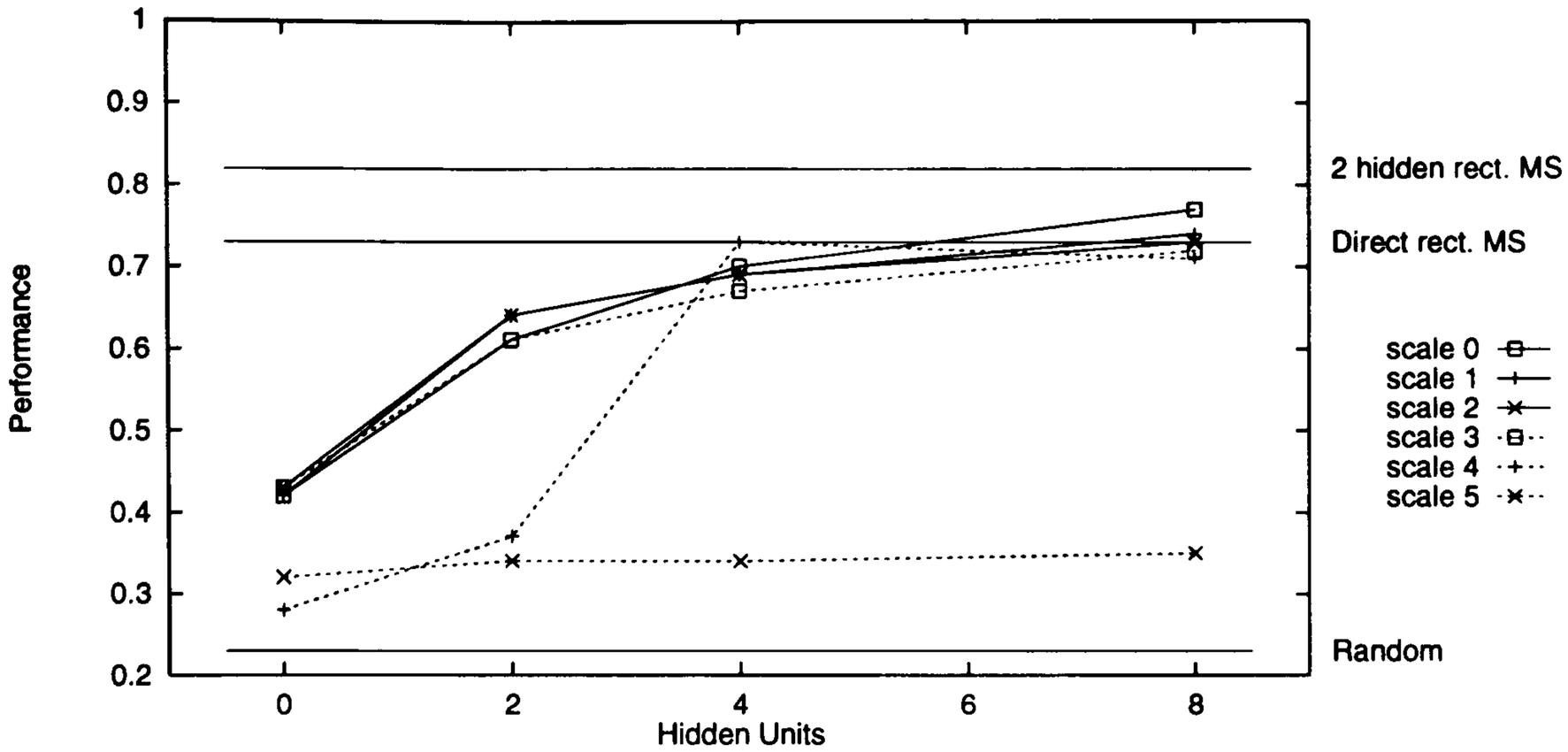


Figure 5.19: Performance as a function of network size for animats with rectified single scale coding. Each scale is one of those used in the multiscale coding condition (scale 0 is the finest; scale 5 the coarsest). Also shown for comparison are the performance of the random animat and rectified multiscale coding animats. Each data point is the mean performance of 3 animats. All standard errors $\leq$0.04, except for scale 4, where standard errors $\leq$0.06.

around the level of the direct rectified multiscale animat with 4 or 8 hidden units. The 4 finest scales (0–3) all show very similar performance, achieving near the level of the direct rectified multiscale animat with 4 hidden units, and a slight increase with 8 hidden units. The best performance of 0.77, by 8 hidden unit animats at scale 0, is significantly lower than the 0.82 performance of the 2 hidden unit, rectified multiscale animats (t-test: $t=2.496$, $p<0.05$). However, since the performance of the finer scale animats increases somewhat from 4 to 8 hidden units, it may be that with more hidden units their performance could match that of the 2 hidden unit rectified multiscale coding animats. This could be easily tested with further simulations.

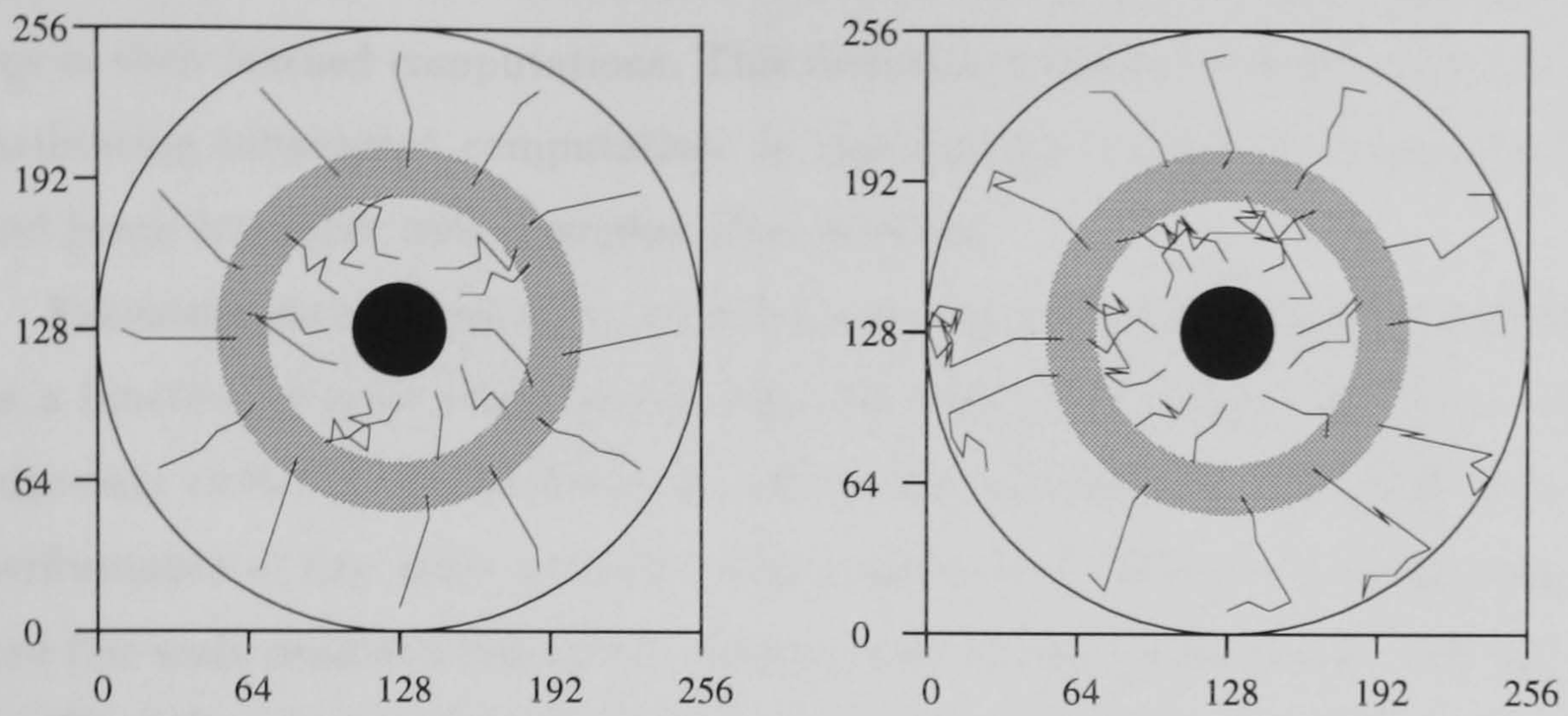
It requires 4 hidden units for rectified single scale animats to achieve the same performance level as direct, rectified multiscale animats; and 8 hidden units for scale 0 animats to perform near the level of 2 hidden unit, rectified multiscale animats. Thus, rectified multiscale coding of the visual array facilitates subsequent computation in that smaller filter networks are required to learn the mapping to useful output than when the visual array is coded at any single scale.

5.5.3 Behaviour

Examination of typical paths show that the animats with single scale 0 to 4, and 4 hidden units, have learned to move quite efficiently toward the circle from outside the goal ring, but have problems moving away from the circle from within the goal ring. The best of these animats (4 hidden unit, scale 4), show behaviour similar to that of the direct, rectified multiscale animat.

Fig. 5.20 shows typical paths of 8 hidden unit, rectified single scale animats at scales 0 (fine) and 4 (coarse), when tested with the solid circle, and the edge only circle. With the solid circle,

(a) 8 hidden unit, rectified single scale 0 (fine scale).
Solid circle: Performance = 0.77. Edges-only circle: Performance = 0.61.



(b) 8 hidden unit, rectified single scale 4 (coarse scale).
Solid circle: Performance = 0.70. Edges only circle: Performance = 0.42.

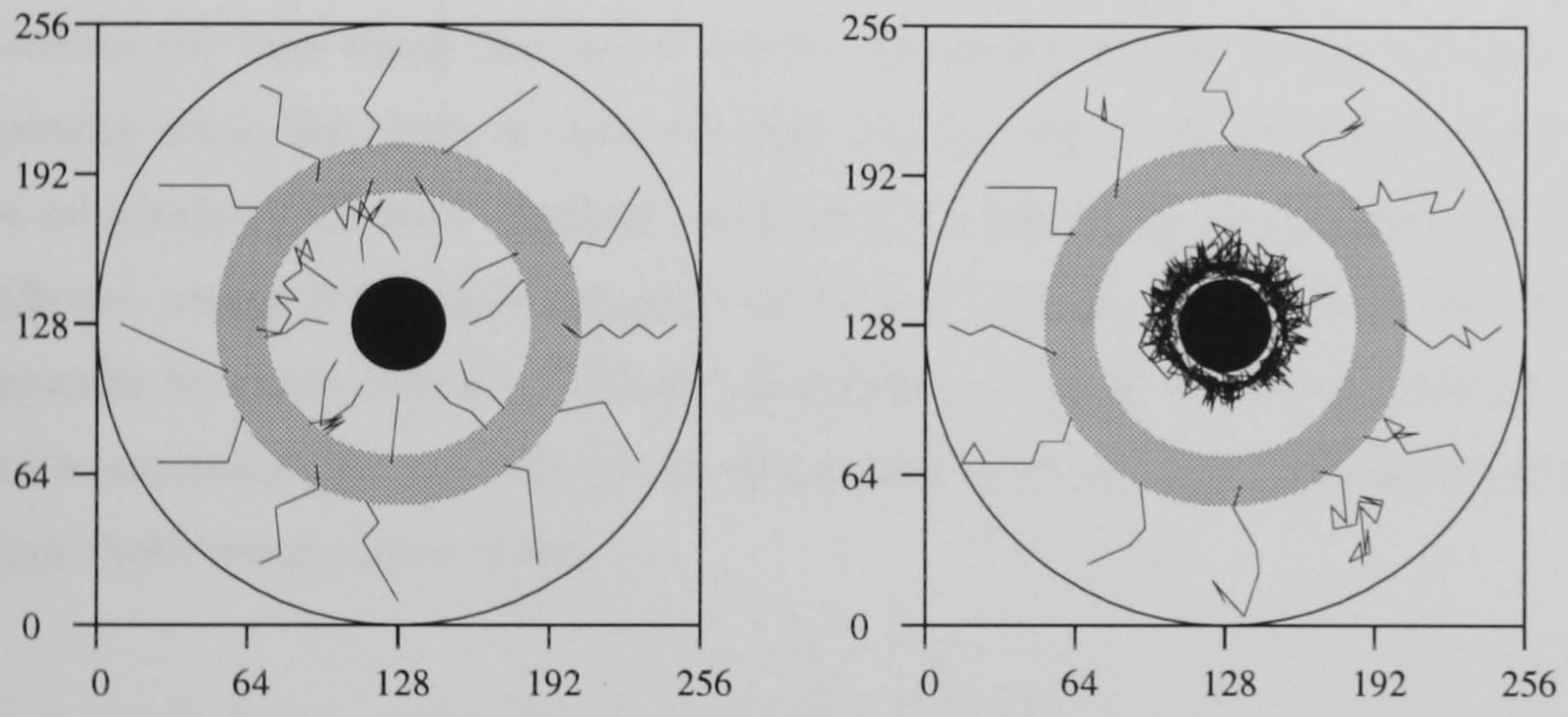


Figure 5.20: Typical paths of 8 hidden unit, rectified single scale animats, tested with the solid circle (as during learning), and the edges only circle. (a) Fine scale. (b) Coarse scale.

the fine scale animat behaves somewhat more efficiently than the coarse scale one. When the solid circle is replaced by just its edges, the performance of the coarse scale animat deteriorates greatly; it just bounces around the circle when inside the goal ring. The fine scale animat is far less sensitive to the replacement, still moving to the goal from both inside and outside the goal ring.

5.5.4 Conclusion

That it requires 8 hidden units for rectified single scale animats to perform as well as direct rectified multiscale animats is an interesting result. It implies that the rectified multiscale code permits a simpler mapping to the required output function than is available at any single scale.

This is supported by the internal structure analysis (fig. 5.12) which showed that direct rectified multiscale animats do use the relation between scales rather than just the pattern of activity at any one in their learned computations. This finding is a further example of rectified multiscale coding facilitating subsequent computation; in this case by making the required computation simpler, and hence learnable with a smaller filter network.

Excepting the coarsest scale, no difference is found in the performance of 8 hidden unit animats as a function of scale when tested with the solid circle image. However, when tested with the edge-only circle, the performance of coarse scale animats is considerably more degraded than the performance of fine scale animats. This implies that different computations underlie the coarse and fine scale animat's behaviour, despite their similar performance with the solid circle.

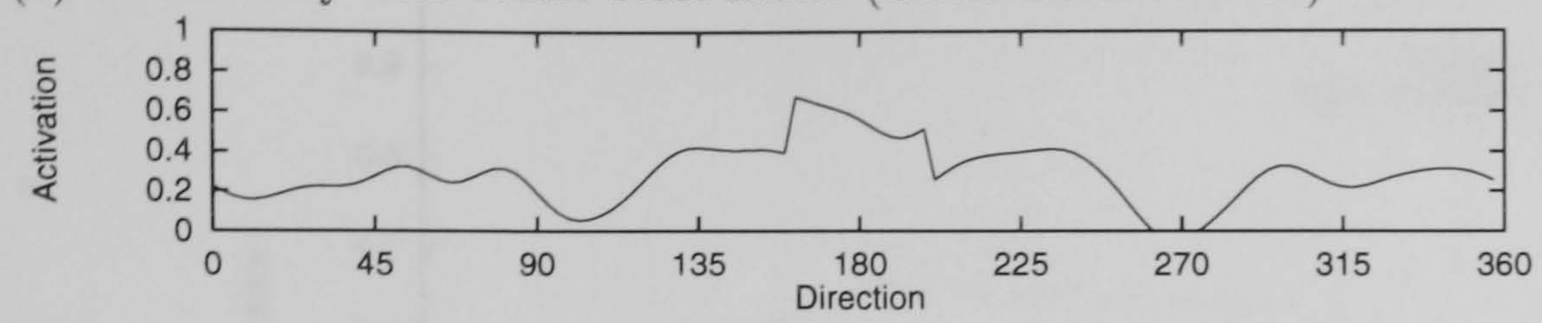
The difference between the pattern of activity caused by the solid and the edges-only image is small at the fine spatial scales, whereas at coarser spatial scales, the difference is more pronounced, with the difference increasing with spatial scale. So the animats that have learned to efficiently perform the task using fine scales will be processing similar arrays to those experienced during learning when the circle is replaced with just its edges. Hence their output and behaviour will be relatively unaffected. Animats that have learned to use coarse scales will be processing very different arrays from those experienced during learning. Hence their output and behaviour will typically be more affected. This is the pattern of results seen here: animats using fine scales to guide movement are less affected by the replacement of a solid subtended angle by just its edges than those using coarse scales.

5.6 Coarse Scale noise

It has been established above that when learning with zero, or independent visual noise, rectified multiscale animats learn to use the relative activity at the coarse and fine spatial scales of the mulitscale array to guide their efficient movement to the goal. Replacing the solid circle with the edge-only circle disrupts the coarse scales of the multiscale array and so considerably degrades the behaviour of these animats; and rectified single scale animats using the coarser scales. Switching to edge-only hardly affects the fine scales of the multiscale array and this is reflected in the finding that rectified single scale animats that use the fine spatial scales are far less affected by the switch.

In the circle approach task of the last chapter, it was shown that when learning with visual noise, animats learn computational solutions involving the scales least affected by noise. In the present case, this suggests that when the noise is coarse scale, and hence rectified multiscale animats are forced to use the fine scales, their performance will be less degraded in the edges-only test than with zero or independent noise. The following simulations test this hypothesis.

(a) Visual array with coarse scale noise. (Circle distance = 60)



(b) Visual array after multiscale filtering

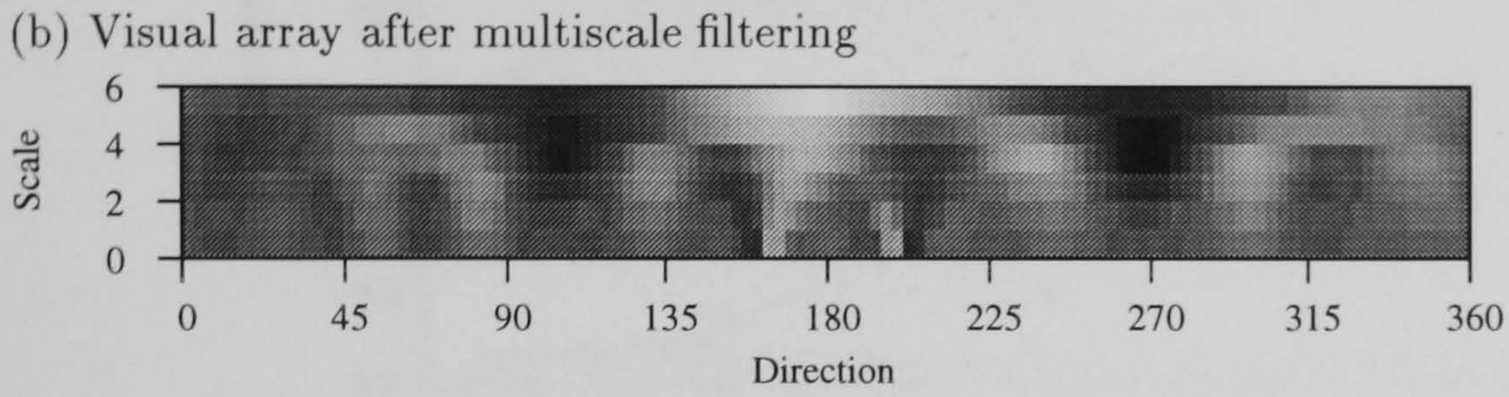


Figure 5.21: (a) Example visual array with coarse scale noise added.
(b) Multiscale convolution of the visual array. In contrast to the activity at the fine scales caused by independent noise, coarse scale noise causes activity at mostly the coarse spatial scales.

5.6.1 Method

Intensity and rectified multiscale coding animats of exactly the same design and parameters as above were simulated with coarse scale noise added to the visual array. The noise has the same statistics as in the previous chapter: independent noise with standard deviation of 0.33 is convolved with a Gaussian of spatial scale 3.0 units to yield coarse scale noise with a standard deviation of 0.1. 3 animats were simulated in each condition, and learning proceeded for 50k trials for animats with 2 or less hidden units, and 150k trials for those with more than 2 hidden units. The extra learning time reflects the finding that with coarse scale visual noise, animats typically took longer to converge than in the noiseless, and independent visual noise cases.

5.6.2 Performance

Fig. 5.22 shows the performance after learning. Intensity coding animats with 2 or less hidden units perform near random, and performance does not get above 0.5 with 12 hidden units. Direct, rectified multiscale coding animats perform at 0.54 and this increases to 0.66 with 4 hidden units. This performance level is not bettered with 8 hidden units.

5.6.3 Behaviour

Fig. 5.23 shows the behaviour of a 4 hidden unit, rectified multiscale coding animat, after learning with coarse scale noise. When tested with the solid circle, behaviour is less efficient than that of animats that learned without noise, but the animat has clearly learned to move both away from the circle when inside the goal ring, and toward the circle when outside of the goal ring. When tested with just the edges of the circle, the animats behaviour is far less degraded than

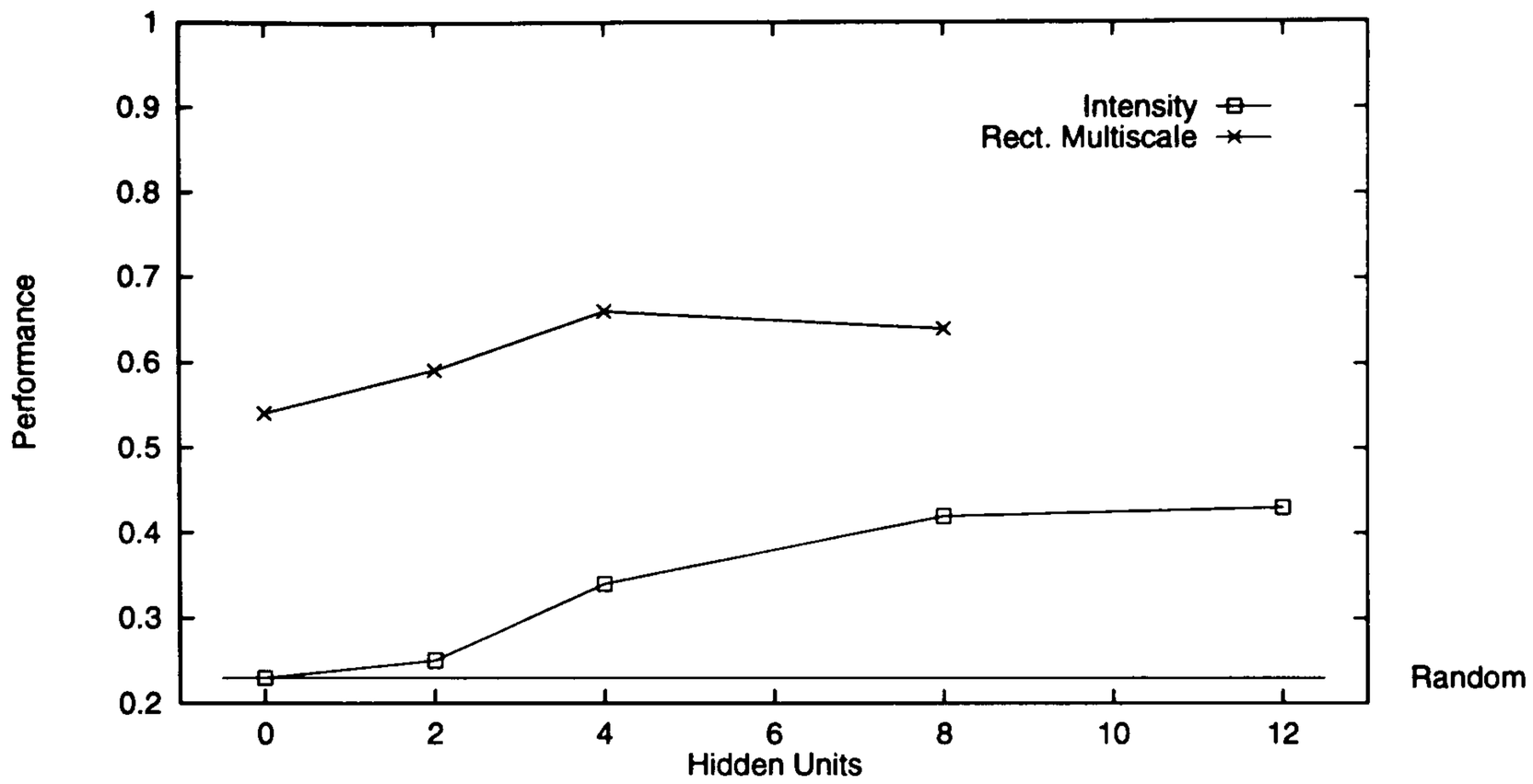


Figure 5.22: Performance as a function of network size for intensity, and rectified multiscale coding learning with coarse scale sensory noise. Each data point is the mean performance of 3 animats. All standard errors$\leq$0.03.

rectified multiscale animats that learned with zero, or independent noise, and this is reflected in the performance scores. Similar behaviour was shown by the other two animats with 4 hidden units, with a mean performance of 0.68, when tested with edges-only and zero noise. This value is significantly higher than the best (fine) single scale animat tested in the same conditions.

5.6.4 Internal structure

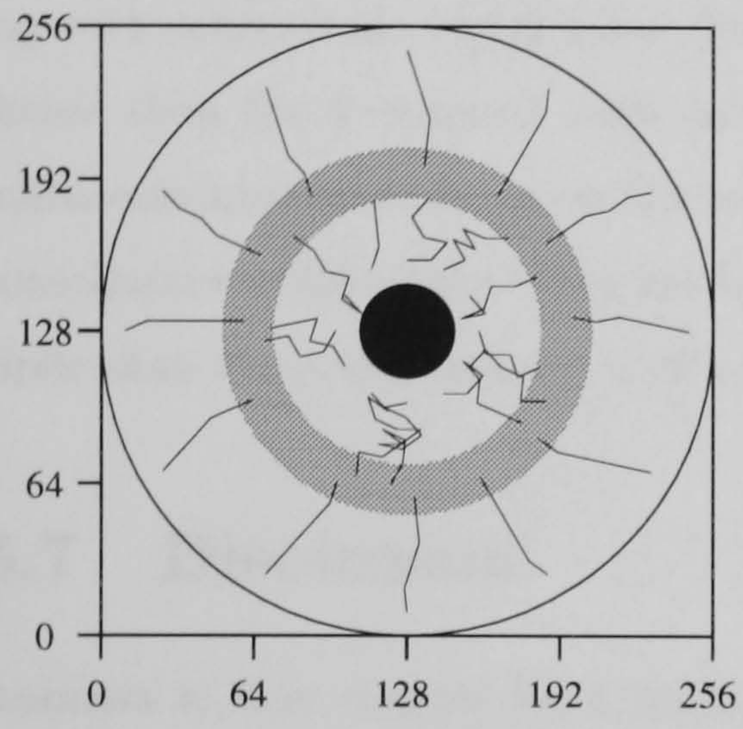
Figs. 5.24 and 5.25 show the weights and response profiles of the 4 hidden unit filter network controlling the animat whose behaviour was shown and discussed above. Comparing this with the corresponding figure for rectified multiscale coding animats in the noiseless case (fig. 5.14), it is clear that the coarse scale noise has led to a very different computational solution. Weights to the 3 coarsest scales are all near zero, implying that animats have learned to use activity at the fine scales to guide movement.

The coarse scale noise has forced the filter network to use the fine scales to compute subtended angle, and this involves estimating the retinal distance between the two localised patches of fine scale activity caused by the circle's edges. In fig. 5.24, hidden units 1 to 3 each respond to a range of subtended angles at localised regions of the receptive field, and between them they approximately tile the central 30 degrees of the networks receptive field. Hidden unit 4 seems to have a complementary function and is active when the circle is out of view, but not when it is around the center of the receptive field.

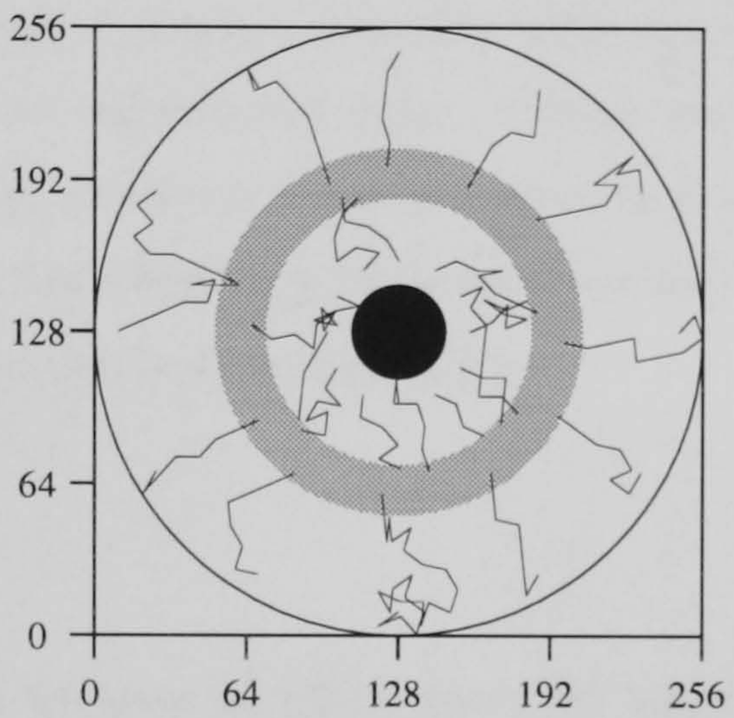
Because the fine scale activity pattern is less modified by the edges-only replacement, there is less impact on these computations than is the case with animats that rely on coarse scale filter activity. This is confirmed by fig. 5.27 which shows the hidden and output unit response patterns

(a) Solid circle, coarse scale noise.
Performance = 0.65.

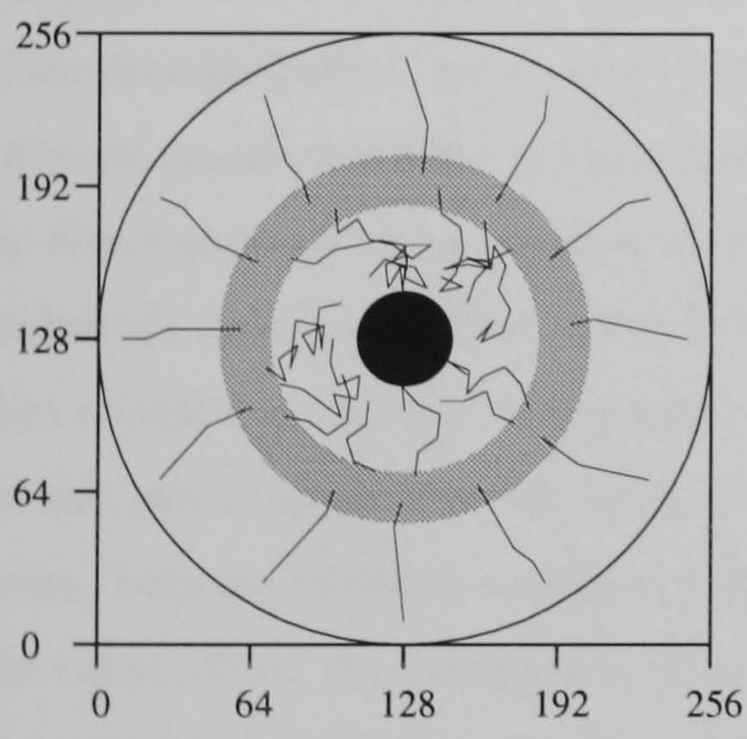
(b) Edges only, coarse scale noise.
Performance = 0.64.



(c) Solid circle, zero noise.
Performance = 0.67.



(d) Edges only, zero noise.
Performance = 0.68.



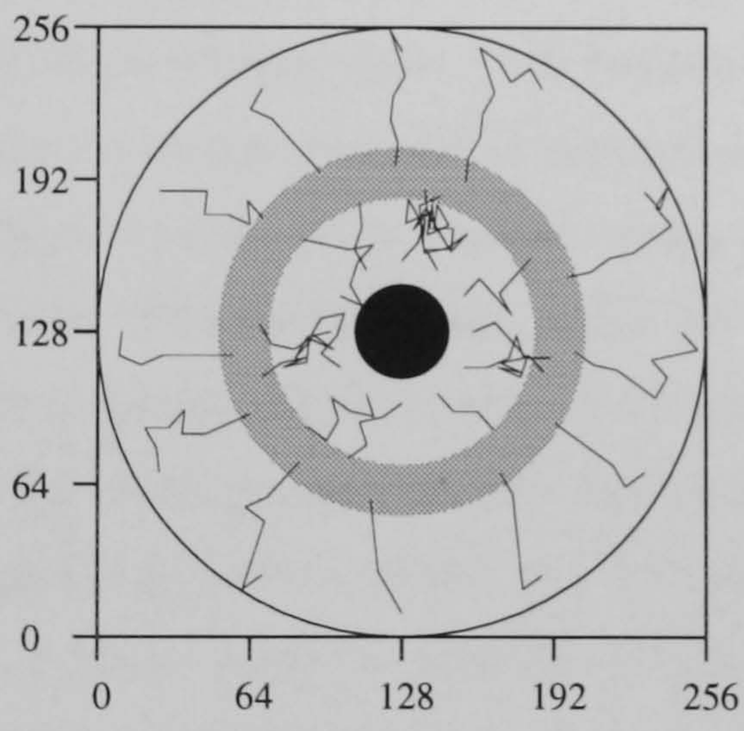


Figure 5.23: Typical paths of a 4 hidden unit, rectified multiscale animat that learned with coarse scale visual noise.
(a) and (b) Behaviour when tested with coarse scale noise at the same level as during learning.
(c) and (d) Behaviour when tested with zero visual noise.

to the edge-only circle. There is less difference between these responses and those to the solid circle than was the case with rectified multiscale animats that learn with zero, or independent visual noise. Hence these animats are far less affected behaviourally by replacing the solid circle with the edge-only circle and behave lake Cartwright and Collett's (1983) honeybees.

5.6.5 Conclusion

Rectified multiscale coding animats considerably outperform intensity coding animats when learning with coarse scale visual noise, though it requires 4 hidden units to achieve peak performance rather than the 2 required with zero or independent noise. Coarse scale noise causes rectified multiscale animats to focus on fine scale activity in order to achieve their efficient performance. A consequence of this is that they are far less affected by replacing the solid circle with the edge-only circle than those that learned with zero and independent noise.

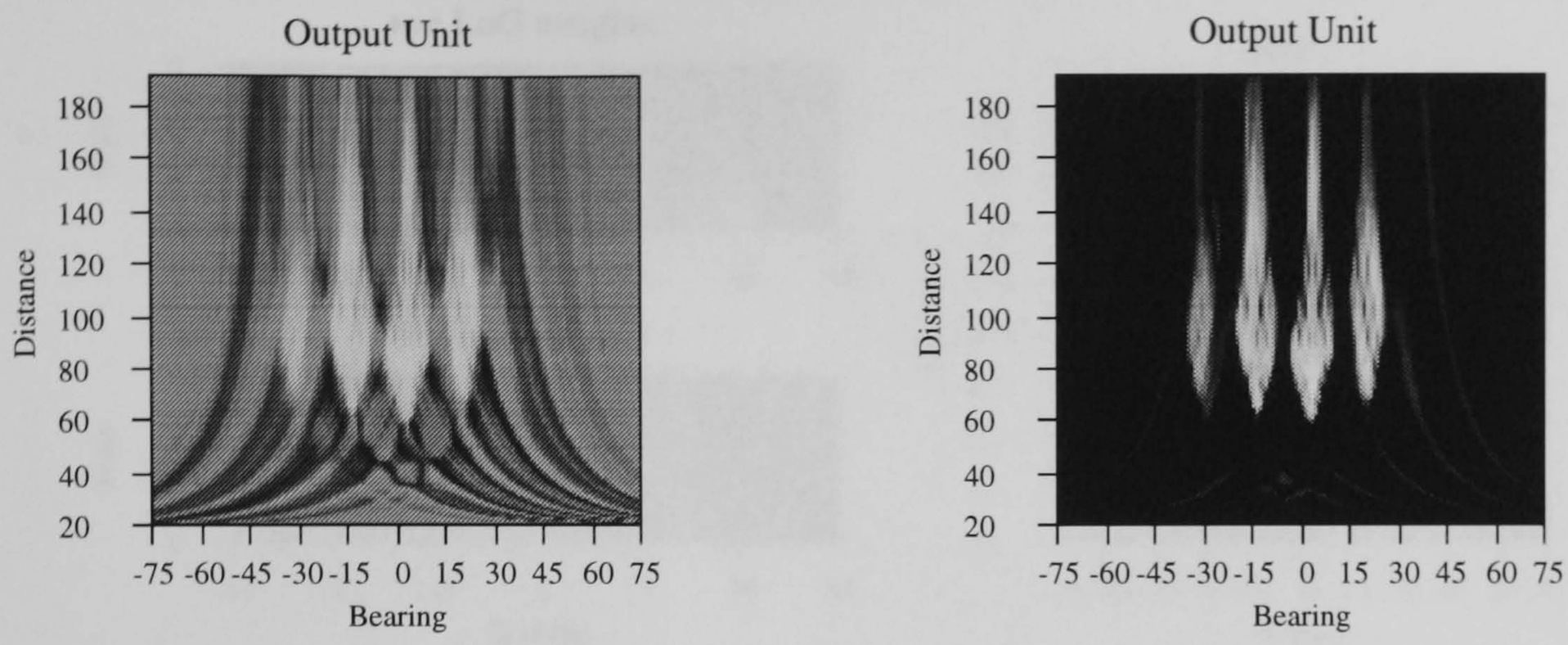
5.7 Discussion

Animats in this chapter have learned to move to where the solid visual angle subtended by a circle falls within a goal range, irrespective of the sign and magnitude of image contrast. Intensity, multiscale and rectified coding lead to animats behaving at a level much better than chance, given enough hidden units. Learning with zero visual noise, both multiscale and intensity coding animats require 8 hidden units to achieve an asymptotic level of performance that that is matched by direct animats using rectified multiscale coding. With 2 hidden units, performance of rectified multiscale coding animats improves further. This finding supports the case of the previous chapter that rectified multiscale coding makes it easier for the filter network to learn the required mapping to the motor array. In the task of the previous chapter, this facilitation occurred only with noise, because rectified multiscale filtering separated activity due to noise from activity due to the circle. Here, the facilitation occurs because rectified multiscale filtering makes the subsequent computations simpler, rather than because it enables multiple computational routes to the output function.

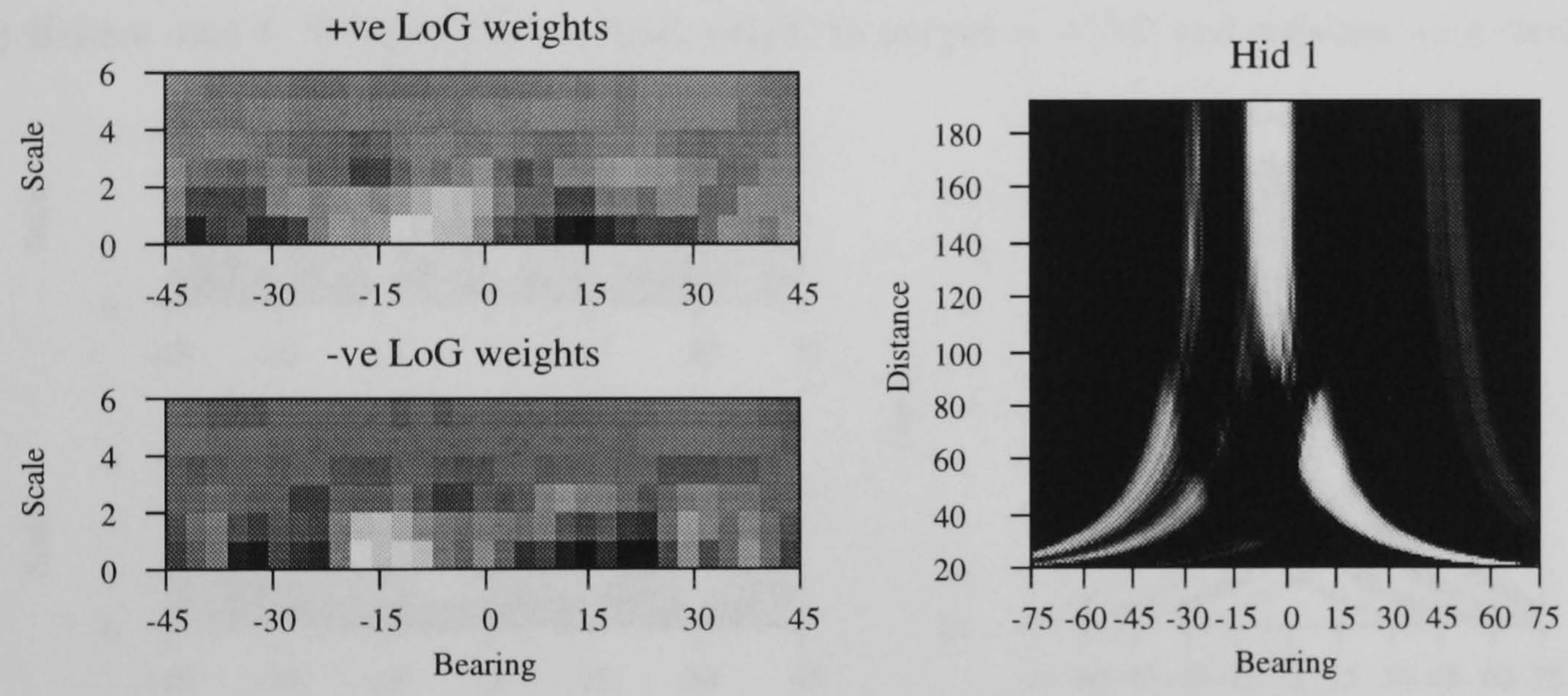
The computational problem that filter networks have to learn to solve in order to efficiently perform the present task is sensitivity to subtended angle, irrespective of the randomly varying contrast. At minimum, they must output a large value when the circle subtends an angle smaller than the goal angle, and a low value when it subtends an angle larger than the goal angle. From the raw visual array, responding to subtended angle, irrespective of contrast, is computationally difficult. This is reflected in the finding that 8 hidden units are required for intensity coding animats to perform efficiently, though at a lower level to rectified mulitscale coding animats.

Rectified multiscale filtering of the visual array makes the subtended angle much more explicit

(a) The response of the output unit to a circle: Raw and thresholded at null activation



(b) Hidden unit 1: Weights (bias = -3.08, weight to output = -2.72) and response to a circle.



(c) Hidden unit 2: Weights (bias = -3.23, weight to output = -2.73) and response to a circle.

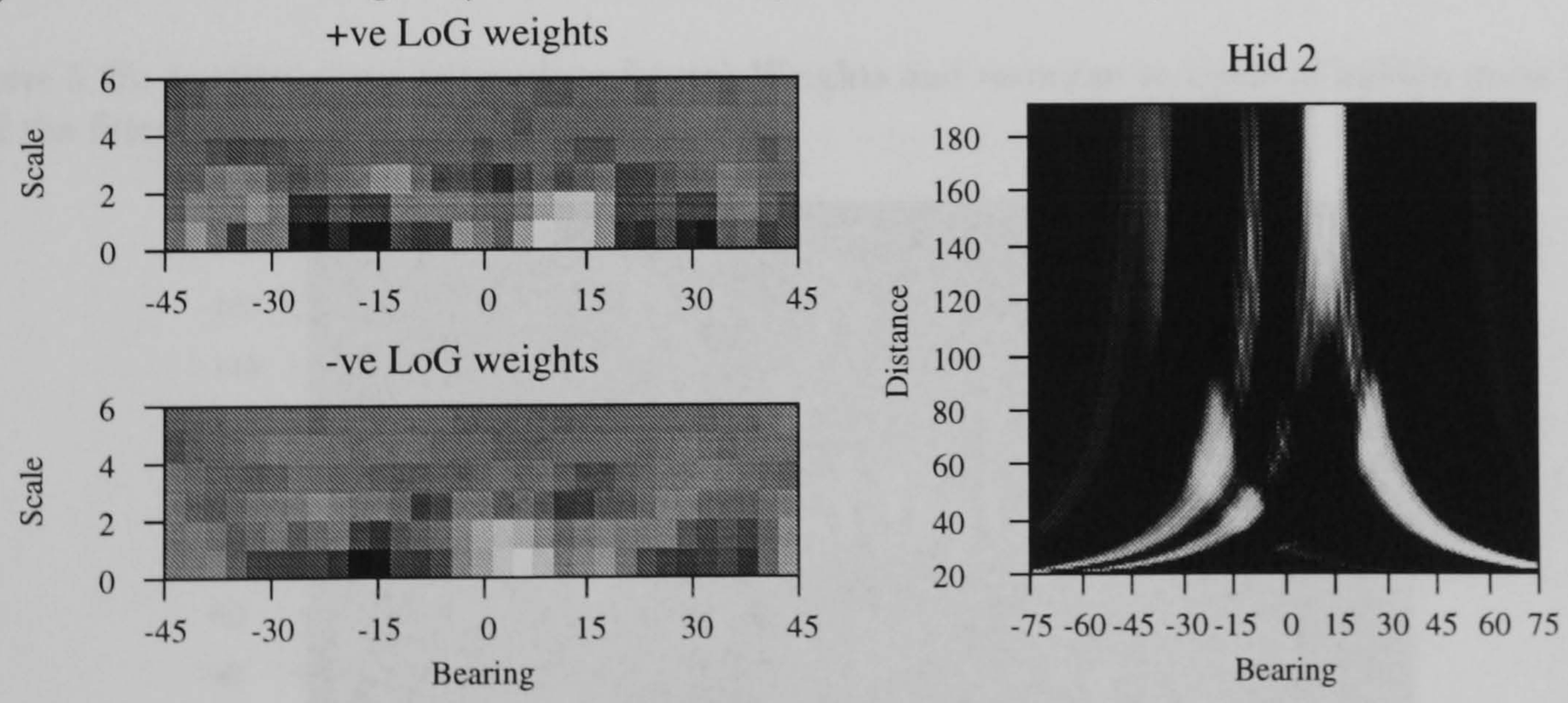
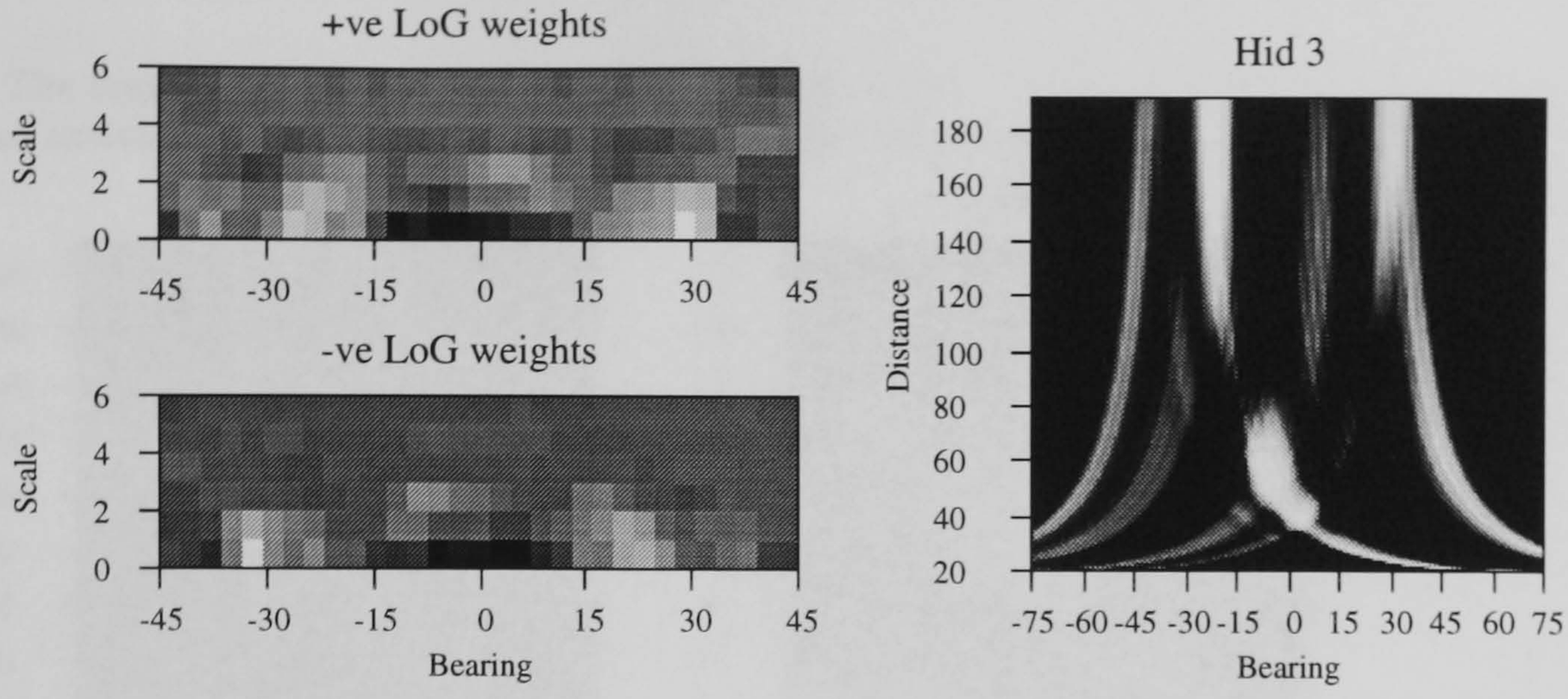


Figure 5.24: Learned weight structure and response pattern of a rectified multiscale coding, 4 hidden unit filter network that learned with coarse scale sensory noise (animat performance = 0.65). (a) Response of output unit. (b) and (c) Hidden units 1 and 2 : Weights (left column), and response to circle, radius 20.
(For these plots : Circle intensity = 0.7, wall intensity = 0.2.)

(d) Hidden unit 3: Weights (bias = -3.29, weight to output = -2.56) and response to a circle.



(e) Hidden unit 4: Weights (bias = 3.50, weight to output = -2.32) and response to a circle.

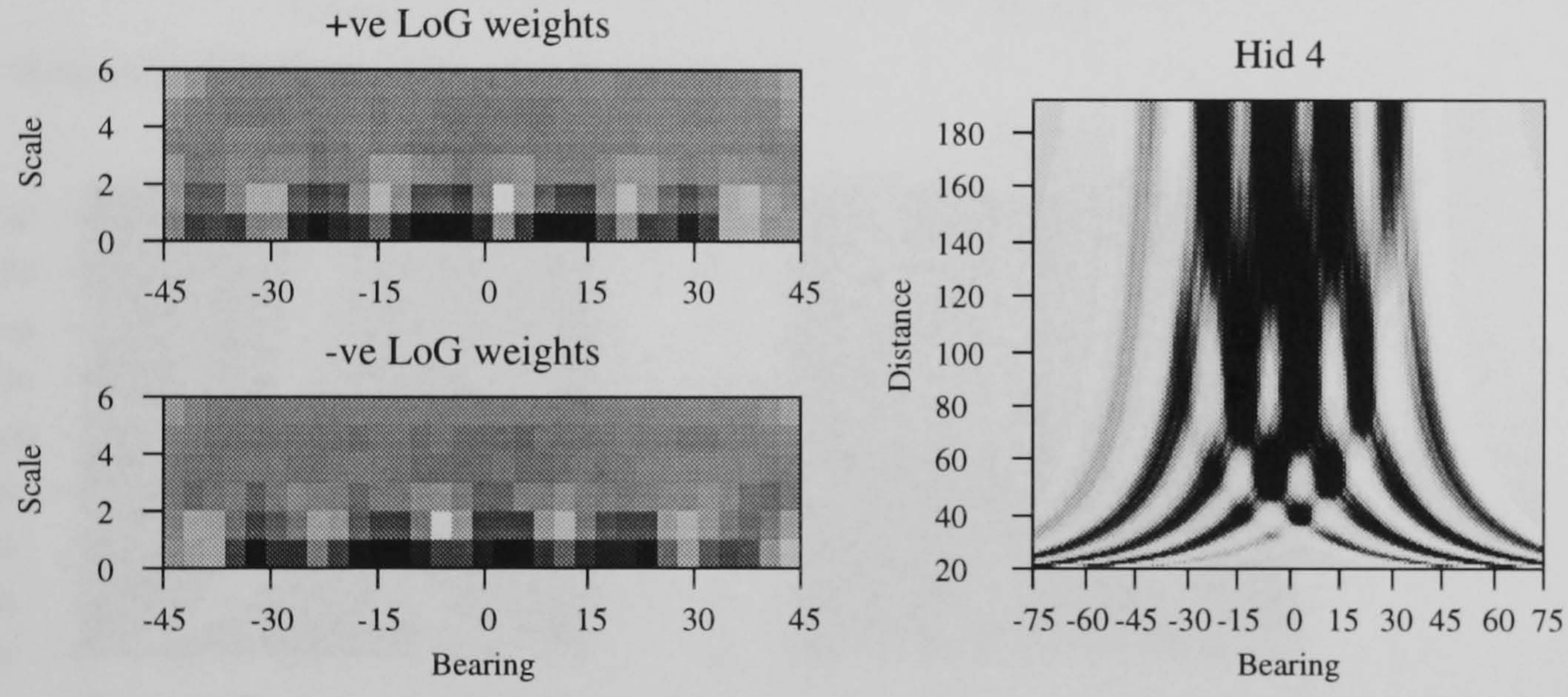


Figure 5.25: (continuation of previous figure) Weights and response to circle of hidden units 3 and 4 of the filter network from the previous figure.

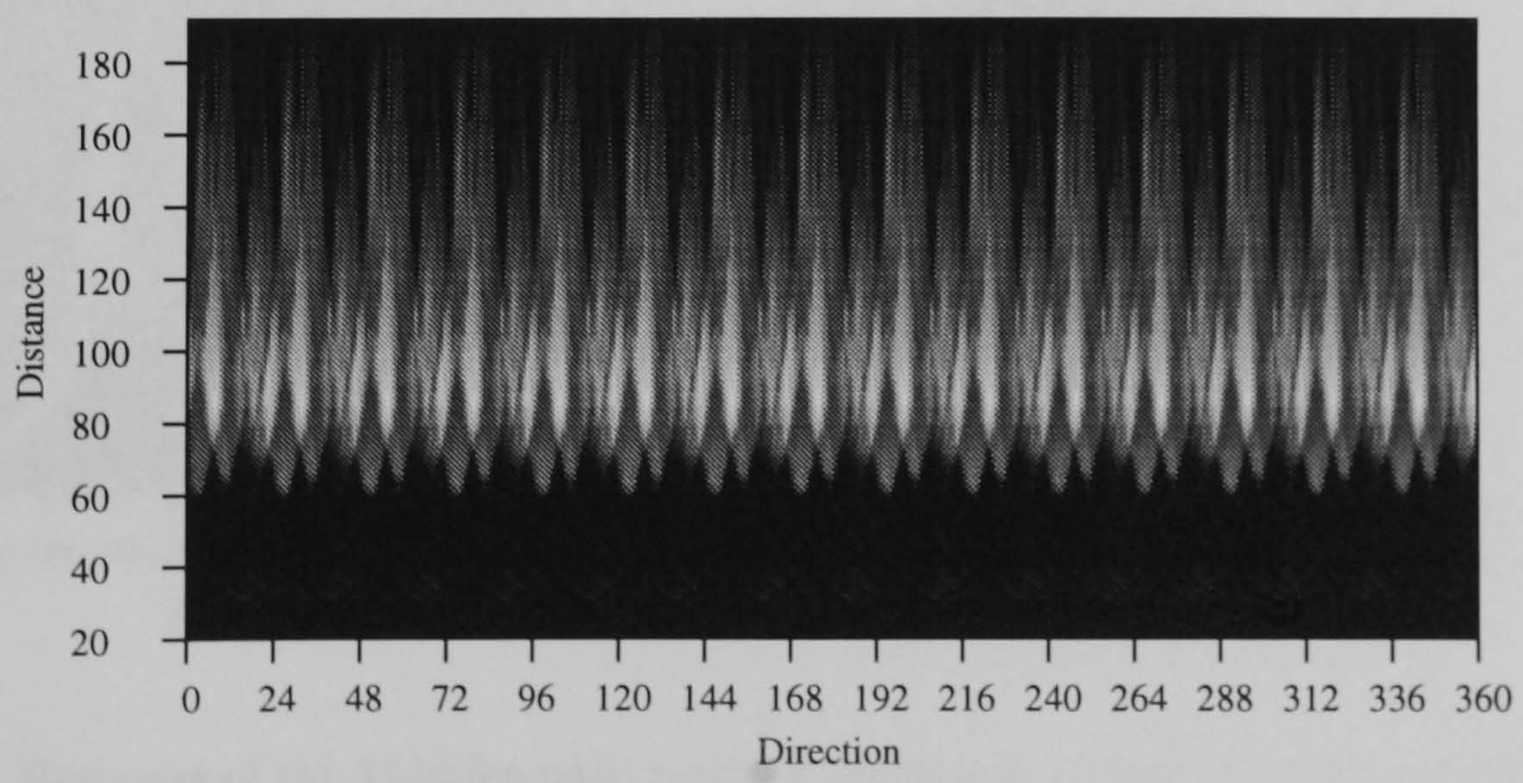
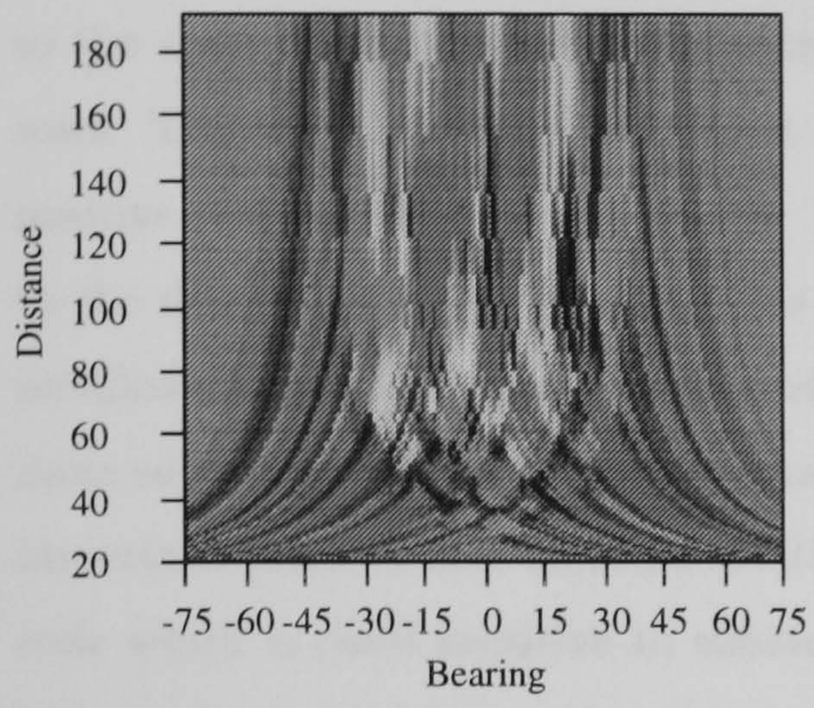
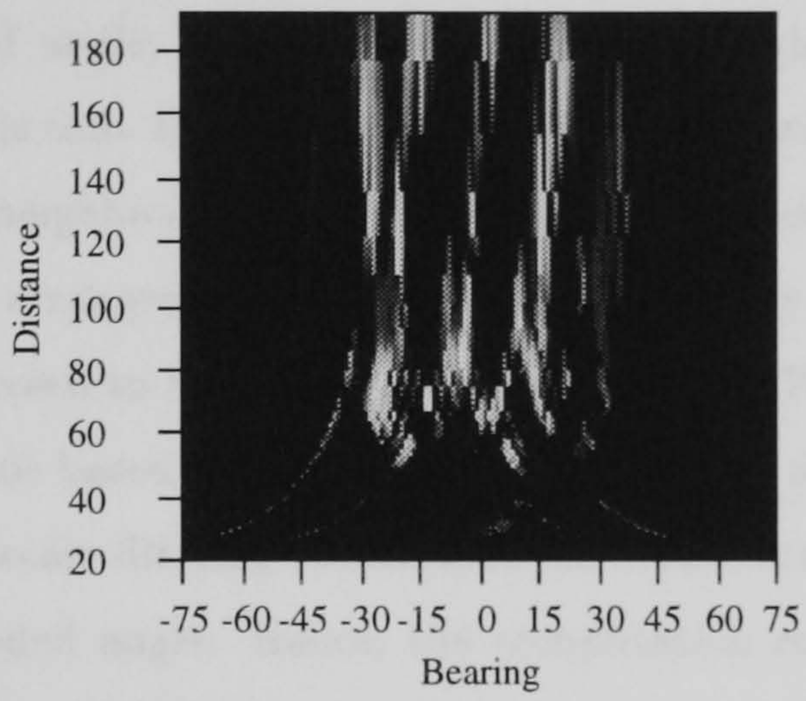


Figure 5.26: Response of the 4 hidden unit rectified multiscale animat that learned with coarse scale noise to the circle at all distance and directions. Response is plotted irrespective of filter network direction and thresholded at the null activation.

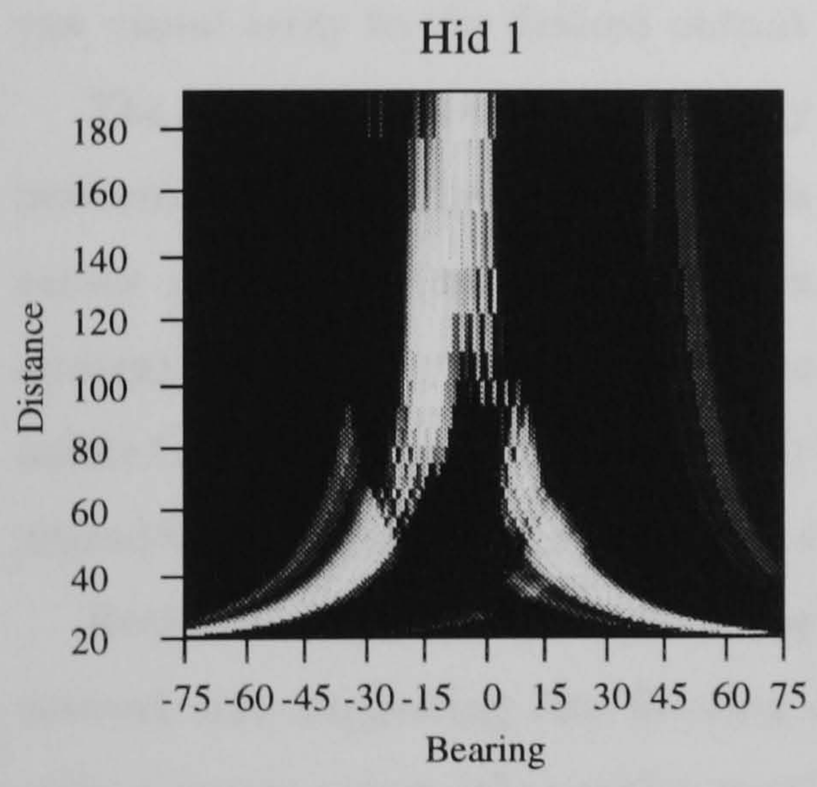
a) The response of the output unit to a edge only circle:
Raw activation

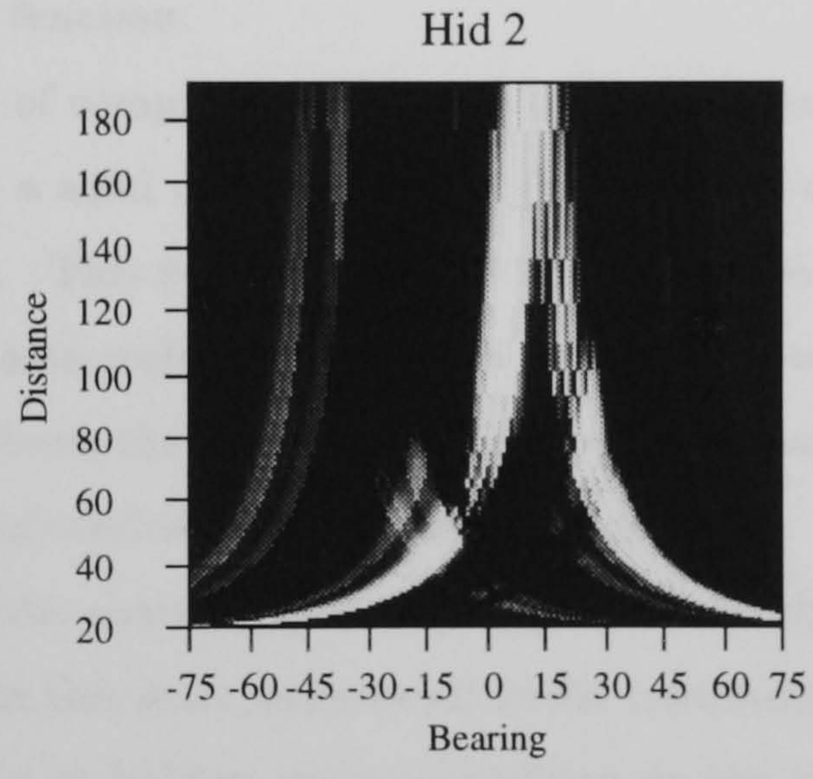


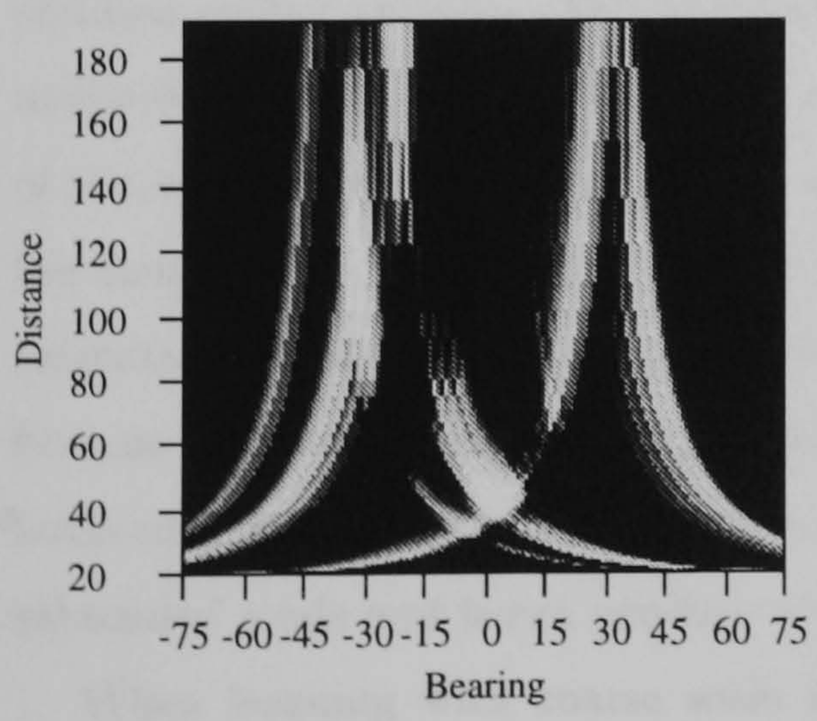
thresholded at null activation.



b) Response of hidden units to edge only circle.







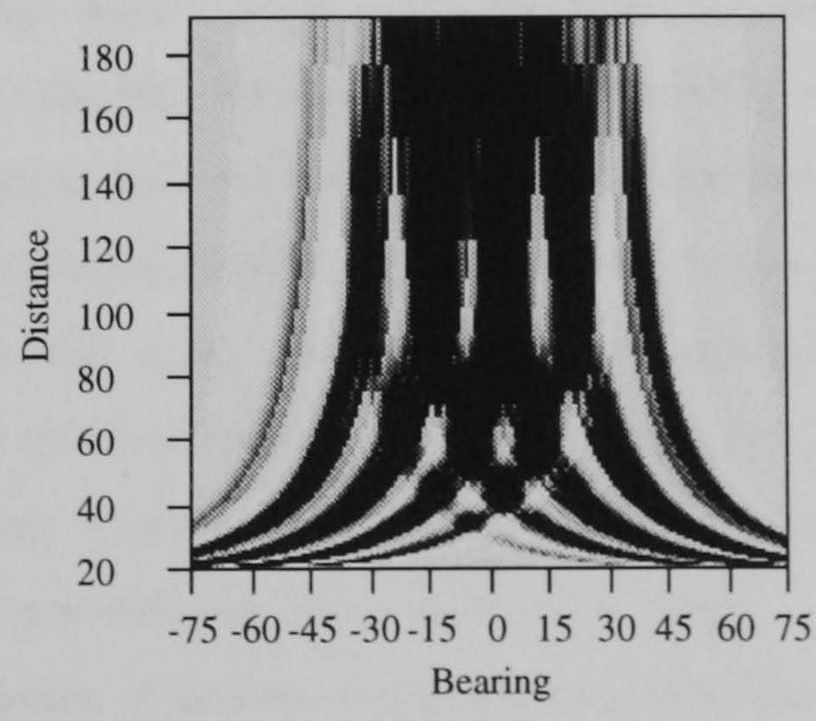


Figure 5.27: Response of the 4 hidden unit, rectified multiscale coding filter network of the previous figure to the edge only circle.

because each filter, through its scale parameter, responds most strongly to a particular range of solid subtended angles. Hence, as the angle subtended by the circle changes, the row of the rectified multiscale array with maximum activity changes. In order to be sensitive to a particular subtended angle, direct filter networks have only to place large positive weights on the row corresponding to the scale nearest the goal subtended angle, and balancing, negative weights on neighbouring rows. The result of this arrangement is that the output unit will respond when activity on the positive row exceeds activity on the neighbouring rows; ie when the subtended angle is close to the desired subtended angle. This arrangement of weights is structurally simple, and direct rectified multiscale coding have been shown to learn an approximation of it. With 2 hidden units, more subtle discriminations can be made based upon the relative activity at different scales. The important point is that rectified multiscale filtering transforms the visual array to an expanded code which is itself sensitive to subtended angle. Hence, the computation required to map this code to the desired output function is far simpler than the computation required to map from the raw visual array to the desired output function.

The simple computational strategy of comparing activity at different scales results in efficient performance when the circle subtends a solid angle. However, in the edges-only test, the circle causes activity at just the fine scales. This severely impairs the above animats because their strategy is based on activity at the coarse scales responsive to the solid subtended angle. Thus, unlike Cartwright and Collett's (1983) bees, the behaviour of rectified multiscale animats is highly degraded by replacement of the solid subtended angle by just its edges.

Rectified single scale animats using the coarsest spatial scale perform poorly regardless of filter network size, suggesting that filtering at this scale removes all useful information. Rectified single scale animats using other scales require 8 hidden units to perform at the same level as direct rectified coding animats. This is because with a single scale, the filter network obviously cannot utilise the relative activity at different scales like the rectified multiscale filter network can. Details of the single scale activation pattern must be used and this greatly increases the complexity of the computation. Replacement of the solid subtended angle by just its edges affects coarse scale animats far more than those using the fine scale, though both are degraded. This is expected because the solid and edges-only circle produce similar arrays after fine scale filtering: fine scale LoGs only respond to edges in either case. Coarse scale LoGs in contrast only respond to the solid subtended angle and hence produce a very different array in the two cases.

When learning with coarse scale noise, 4 hidden units are required for rectified multiscale animats to acheive a peak performance which is considerably better than that of intensity coding. As in the previous chapter, coarse scale visual noise causes more problems for intensity coding than independent noise. Examination of the internal structure of rectified multiscale coding animats shows that they learn to respond to the circle using the fine scales, because they are the least

effected by noise. A consequence of this is that, like honeybees, their behaviour is relatively unaffected by replacement of the solid subtended angle by just it's edges.

Chapter 6

Learning multiple subtended angles

6.1 Introduction

In the previous chapters of this thesis, there was a single spatial goal within the environment. Hence, in order to successfully perform the task, animats had to learn to map the visual array at each location within the environment, to a motor array coding for movement leading the animat nearer to that goal. At each location there is a unique direction pointing toward the goal, and animats must learn what it is. Animals, in contrast, can not only learn to steer toward one spatial goal at a time, but concurrently learn to steer to each of a number of spatial goals; switching between them as the occasion demands. For example, honeybees can learn to steer toward several foraging sites at the same time (Collett, 1992). This chapter addresses this issue by extending the simulations of the last chapter so that the task becomes concurrently learning to move to each of a number of spatial goals.

In the last chapter, the task for the animats was to move to a location where the angle subtended by the image of a circle fell between a certain range. Animats were shown to be able to learn the task, with rectified multiscale coding leading to superior performance than intensity coding. Here, instead of a single goal range of subtended angle, animats must learn to move to each of a number of goal ranges of subtended angle. On any particular trial, one of this number is randomly chosen as the current goal. Animats are informed as to which is the current goal by means of an additional input, and they receive positive reinforcement only when entering the current goal region.

In the simulations reported here, with a circle of radius 20 and an arena of radius 192, there are four, non-overlapping goal regions as shown in fig. 6.1. During learning, one of the four goal

regions is chosen at random on each trial. This is input to animats by setting one of four extra input units to 1 and the rest to zero. Note that one of the four goal regions (number 2 in fig. 6.1) is same as the single goal region of the previous chapter. Hence, the task in the previous chapter is a behavioural subtask of this chapters task in that competence at moving to any of a number of goal regions implies competence at moving to any individual one.

The difficulty for reinforcement learning in this task is that for a large region of the environment (from distances of between 56 and 128 from the circle), the animat must move toward or away from the circle depending upon the current goal. Two opposite behaviours are required for each of these locations; which of the two behaviours should be acted is determined by which is the current goal. Hence, there is not a single utility function that will lead to competent behaviour. The utility function learned by an animat competent at this task will itself be a function of the current goal. This is shown to be the case in section 6.5, where the response of an animat after successful learning is analysed in detail.

In chapter 4, rectified multiscale coding was shown to lead to superior performance than intensity coding in the presence of noise. The performance of multiscale coding (without the rectification step) animats was shown to fall between these two. In chapter 5, rectified multiscale coding was again shown to lead to superior performance, with a smaller number of hidden units, than intensity coding. Again, the performance of multiscale coding fell between these two. Given these results, and the similarity in the environment of the present and previous tasks, only intensity and rectified multiscale coding animats are compared here. Based upon the previous results, it is assumed that multiscale coding animats would learn to perform at a level above that of intensity coding animats, but below that of rectified multiscale coding animats.

6.2 Processing

Processing is as in the last two chapters, except that in addition to the weights to the sensory input, each hidden unit has a weighted connection to the four additional units that specify which is the current goal (see figs. 6.2). In the case of a direct network, the output unit has weights to these four units. They are processed exactly as sensory input units. As in the previous two chapters, the visual array, or the rectified multiscale array, is then convolved by the filter network to yield a 15 element motor array. The goal units have the same activation levels for each filter network (see fig. 6.3 and fig. 6.3).

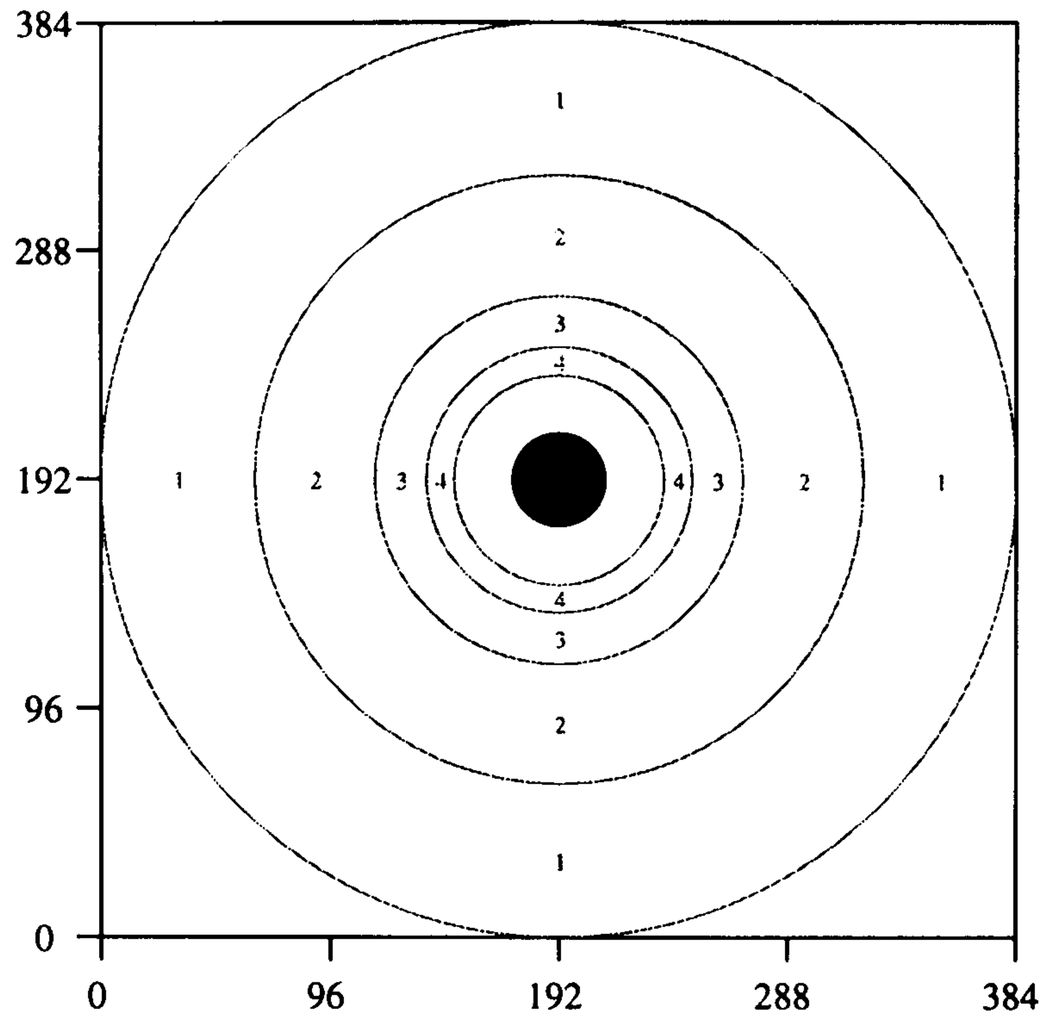


Figure 6.1: The environment is a circular arena of radius 192, empty except for a solitary circle, of radius 20. The invisible goal region on each trial is chosen randomly from one of 4 non-overlapping regions, each specified by a range of subtended angle.

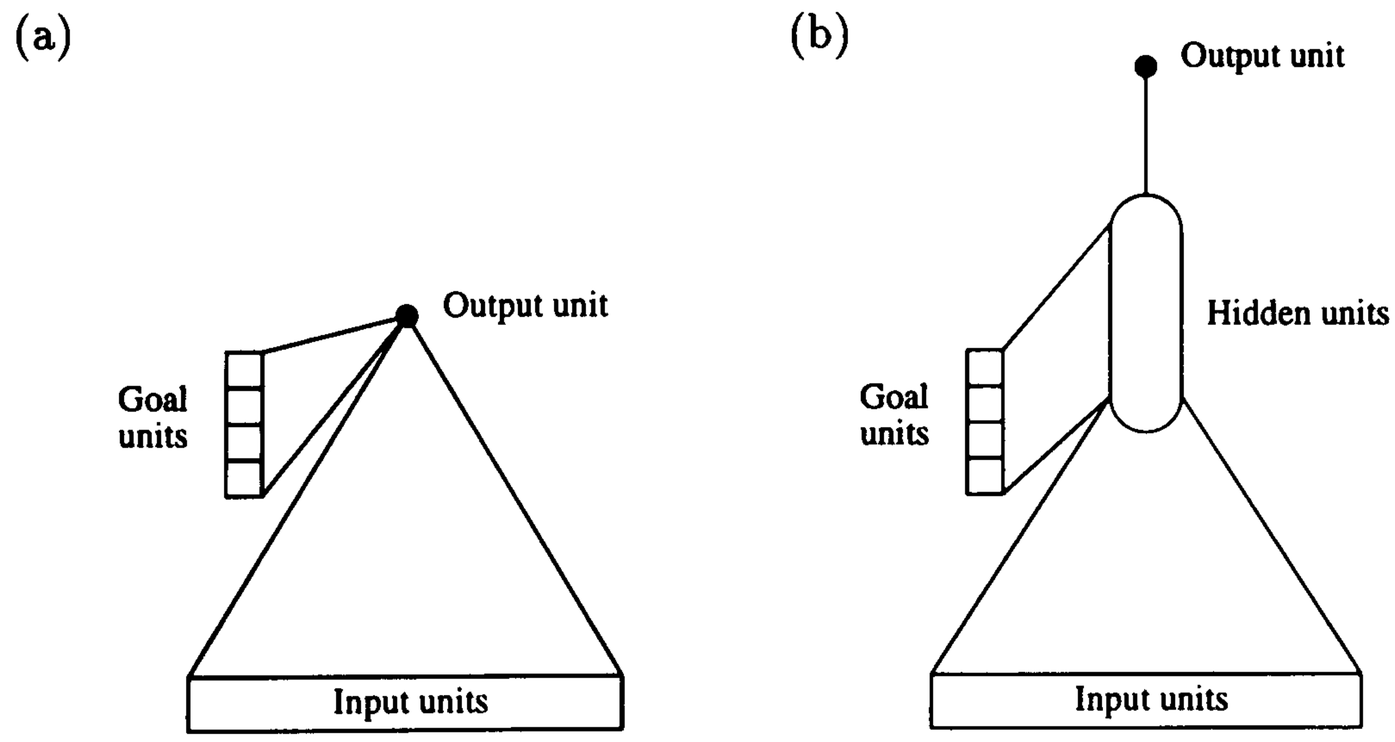


Figure 6.2: Filter networks. In addition to the sensory input, hidden units (or the output unit in the case of direct networks) have weighted connection to four binary goal units.

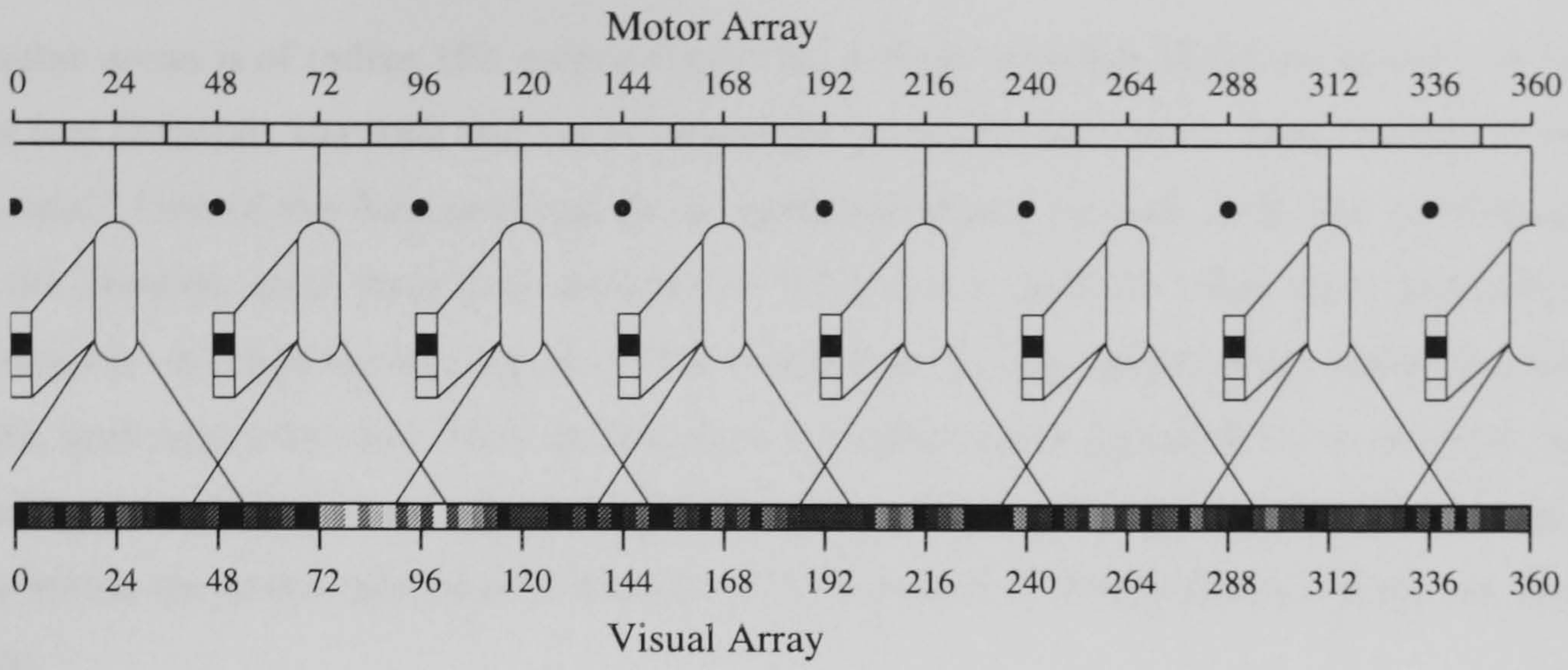


Figure 6.3: Animat sensorimotor control in the intensity coding condition. The visual array is convolved by the filter network which also has input from the goal units. Black dots mark undrawn filter networks.

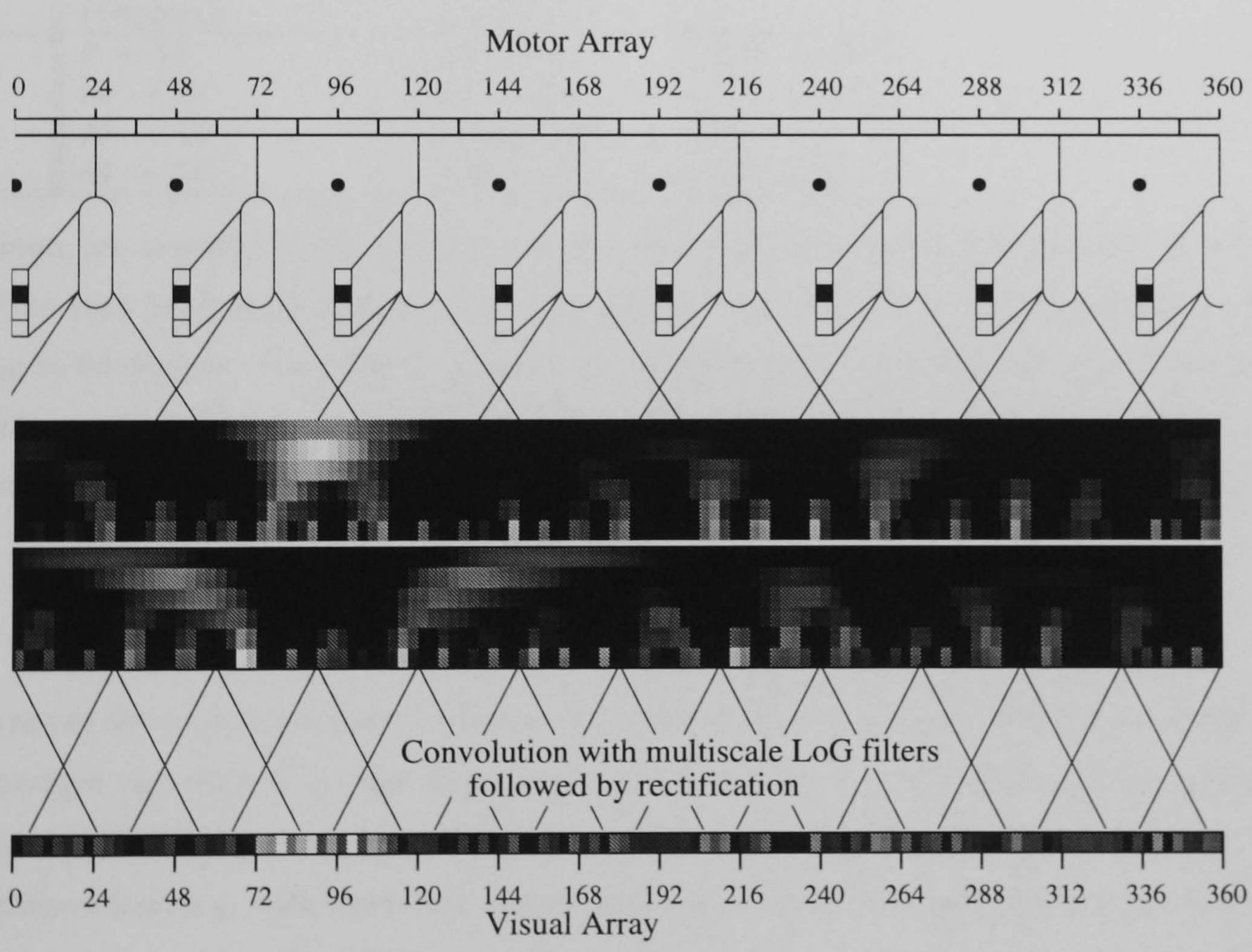


Figure 6.4: Animat sensorimotor control in the rectified multiscale coding condition. The visual array is convolved with multiscale LoGs and then rectified as per the previous chapters. The rectified multiscale array is then convolved by the filter network which also has input from the goal units. Black dots mark undrawn filter networks.

6.3 Simulation Method

The circular arena is of radius 192, empty except for a circle of radius 20 in the center. As in the previous two chapters, the wall and circle intensities are randomly chosen from between 0 and 1 on each trial. One of the four goal regions is randomly chosen on each trial, the corresponding unit in the animats input layer goal units set to activation 1, and the other three goal units set to activation 0. A reinforcement signal of 0 is received on all time steps except when the animat enters the goal region for that trial. In this case, a reinforcement signal of 1.0 is received, and a new trial begun. Trials last a maximum of 500 time steps. Animats start each trial at a random location within the arena, and move a distance of 10 in one of 15 evenly spaced directions on each time step.

The following table lists the subtended angle range for each of the four possible goals. Also listed are the distances from the circle, radius 20, corresponding to these subtended angle ranges; and the number of receptors activated by an image subtending the angles (the 120 sensors each have a resolution of 3 degrees).

Goal	Subtended Angle range (Degrees)	Num receptors activated	Distance
1	$6 \rightarrow 18$	$2 \rightarrow 6$	$128 \rightarrow 192$
2	$18 \rightarrow 30$	$6 \rightarrow 10$	$77 \rightarrow 128$
3	$30 \rightarrow 42$	$10 \rightarrow 14$	$56 \rightarrow 77$
4	$42 \rightarrow 54$	$14 \rightarrow 18$	$44 \rightarrow 56$

Animats are simulated with neural network controllers with either a single layer of weights (direct), or with 2,4,8 or 12 hidden units. All networks have a fan in of 31 receptor units, corresponding to 93 degrees. The receptor array is either intensity, or rectified multiscale coded, using 6 LoG filters with the same scales as in the previous two chapters. The learning algorithm, and parameter settings of learning rate and momentum, are exactly as in the previous two chapters.

6.4 Performance and behaviour

Fig. 6.5 shows the performance of the animats after 100K learning trials, by which time all animats had converged and showed no sign of increasing performance. A randomly behaving animat in the same conditions has a mean performance of 0.16 ± 0.01. All animats were tested over 1000 trials without learning, with randomly chosen starting positions and goal regions on each trial, and a maximum trial length of 500 steps. The performance level of the random animat is shown in fig. 6.5 for comparison with the performance of animats after learning the present task.

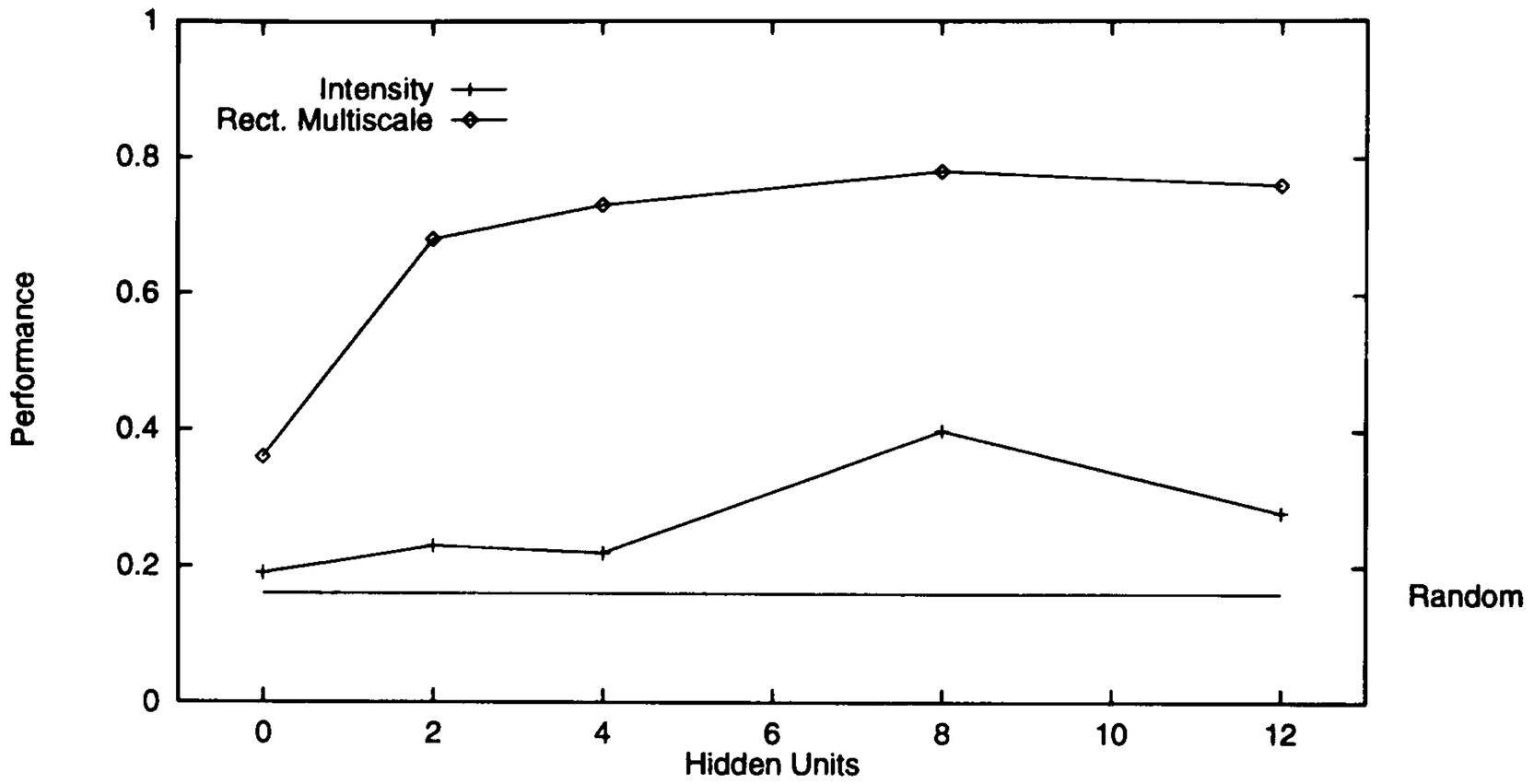


Figure 6.5: Performance as a function of filter network size for intensity and rectified multiscale coding animats learning multiple subtended angles. Each data point is the mean performance of three animats. All standard errors$\leq$0.06 for intensity coding. Standard errors$\leq$0.04 for rectified multiscale coding, except for direct, rectified multiscale where standard error=0.07.

6.4.1 Intensity coding

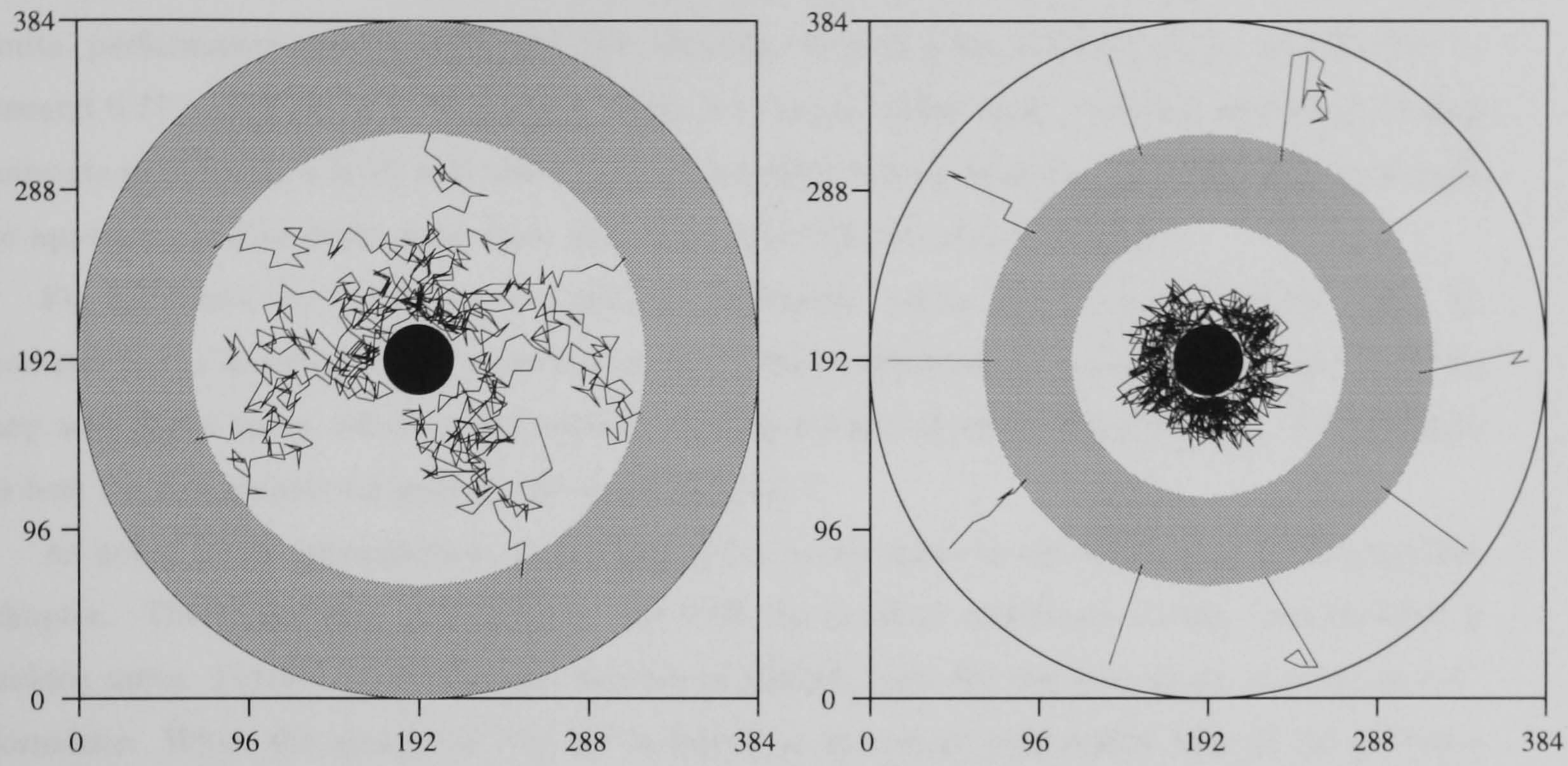
Animats using intensity coding perform poorly. With 4 or less hidden units, performance is little better than that of the random animat. With 8 hidden units, performance rises to 0.4. While this is better than chance, these animats have not learned very much. With 12 hidden units, one of the animats performed at around 0.4 (the same level as with 8 hidden units), and the other 2 at about 0.2, resulting in an average performance of less than 0.3.

Fig. 6.6 shows the behaviour of an intensity coding animat with 8 hidden units, in each of the four goal conditions. For goal 1 (fig. 6.6a), the animat just wanders randomly. For goals 2, 3 and 4 (fig. 6.6b–d), the animat is competent at moving toward the circle when outside the goal ring, but is completely unable to move away from the circle when located within the goal ring. Hence, despite 100k learning trials and 8 hidden units, this animat has only learned to approach the circle.

Other intensity coding animats that achieve a better than random performance also only learn to approach the circle. It should be noted that intensity coding animats have not even learned particularly efficient approach form outside the goal ring. As was shown in the previous chapter, in the noiseless case, intensity coding animats with 4, or more, hidden units are able to learn efficient approach from outside a single goal ring. This implies that the extra computational demands of learning multiple subtended angles seriously degrades their performance on any of the individual tasks.

a) Goal 1.

b) Goal 2.



c) Goal 3.

d) Goal 4.

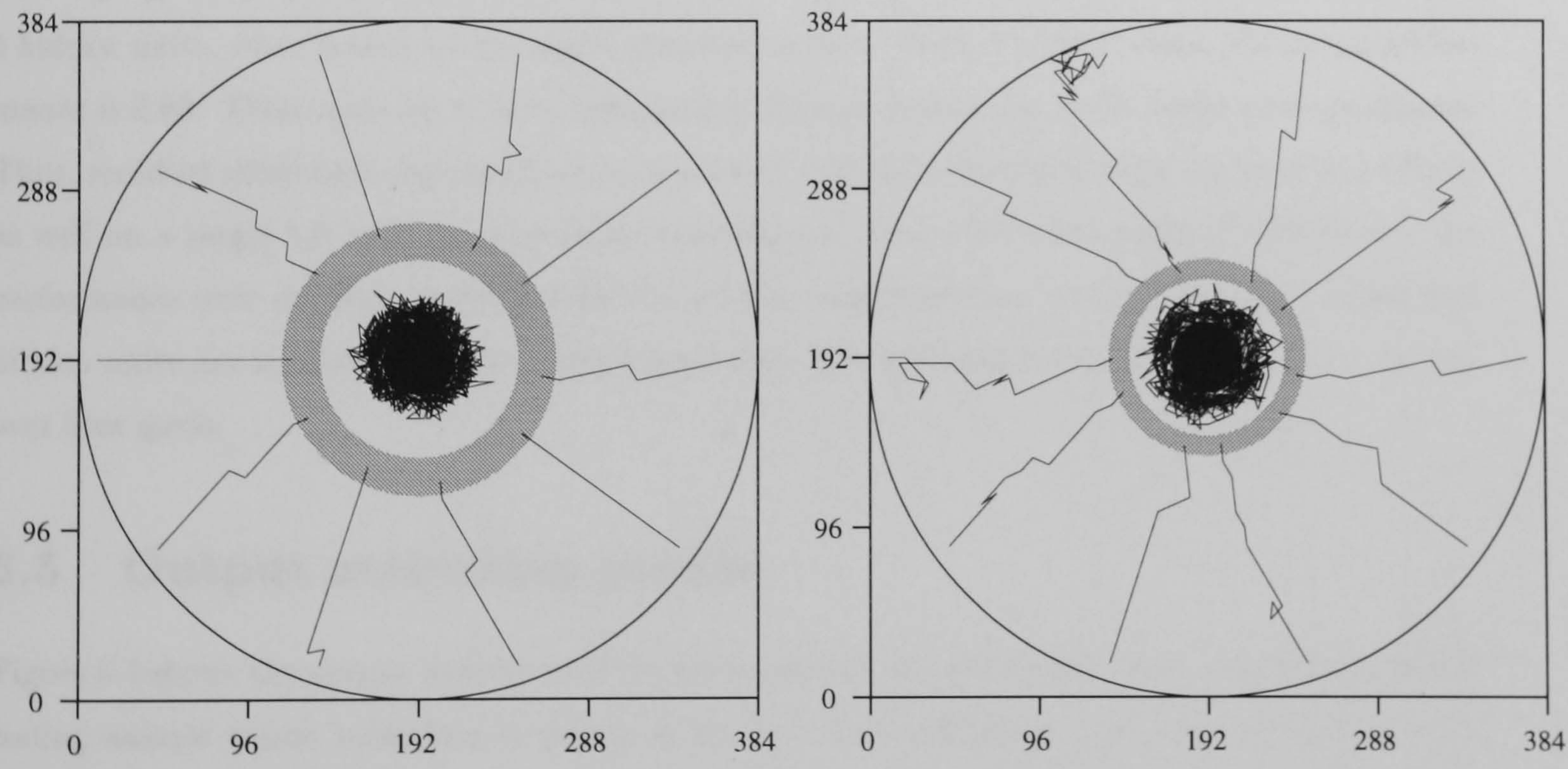


Figure 6.6: Paths of a single intensity coding animat with 8 hidden units. a)–d) show paths from different start locations for each of the four invisible goal regions (shown in gray). The overall performance of this animat is 0.48. In a), animats start near the circle. In b)–d), animats start near the circle for half the trials and at the arena edge for the other half.
For these plots : Circle intensity = 0.7, wall intensity = 0.2.

6.4.2 Rectified multiscale coding

Direct, rectified multiscale coding animats perform at 0.36, considerably better than the random animat, and at around the same level as the best of intensity coding animats. With 2 hidden units, performance rises to 0.68, and this increases to 0.72 with 4 hidden units, and further to around 0.77 with 8 or 12 hidden units. With 2 or more hidden units, rectified multiscale coding animats perform at a level well above that of intensity coding animats. They have learned both to approach, and to move away from the circle, when the occasion demands.

Fig 6.7 shows the behaviour of a rectified multiscale coding animat with 4 hidden units. In contrast to the intensity coding animat, this one is able to efficiently move toward the goal ring from any area of the arena, whether this requires moving toward, or away from the circle. Performance is best for this animat for goal 2, and worst for goal 1.

As noted in the introduction, goal 3 (fig. 6.7c), is the same as the single goal of the previous chapter. There, the best performance was 0.82, by rectified multiscale coding animats with 2 hidden units. Further increasing the number of hidden units did not lead to an increase in performance. When the animat in Fig. 6.7 is tested on the single goal region task of the previous chapter[1], it performs at a level of 0.80, comparable with that of the best of the animats learning the single goal region. The mean performance of the three rectified multiscale coding animats with 4 hidden units, when tested on the single goal task is 0.84. With 8 hidden units, the mean performance is 0.83. These animats achieve comparably high performances in the other goal conditions. Thus, rectified mulitscale animats that have learned multiple subtended angle are able to perform as well on a single one of them as animats learning just that subtended angle. Furthermore, the performance over multiple goals is achieved with a computational economy in that, whilst two hidden units are required to learn a single goal, only four are required to learn to perform as well over four goals.

6.5 Output activation profile

Figure 6.8 shows the output activation of the filter network of the 4 hidden unit, rectified multiscale coding animat whose behaviour is shown in fig. 6.7. The activation is in response to the circle placed at the center of the networks receptive field. Fig. 6.8 shows this activation for each of the 4 goal conditions as a function of the distance of the center of the circle from the animat. This animat has learned to perform well at this task, and this is reflected in the structure of the learned output profile as shown in fig. 6.8 and detailed below.

The horizontal line in each graph of fig. 6.8 mark the filter network's null activation level : the

[1] Done by testing it only in goal condition 3, and with the reduced arena size of the previous chapter

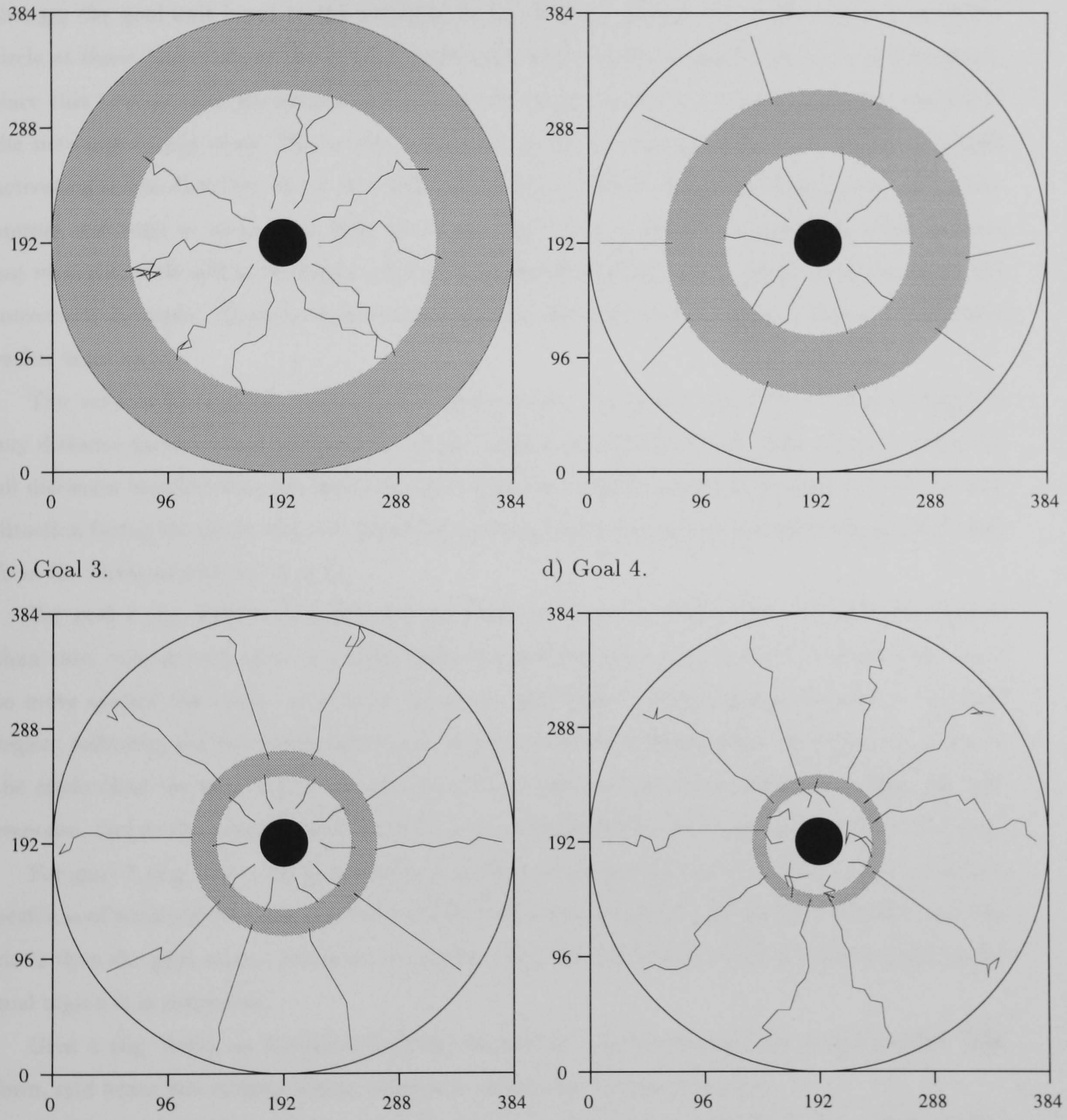


Figure 6.7: Typical paths of a single 4 hidden unit, rectified multiscale coding animat. a)–d) show paths from different start locations for each of the four invisible goal regions (shown in gray). The overall performance of this animat is 0.78. In a), animats start near the circle. In b)–d), animats start near the circle for half the trials and at the arena edge for the other half.
For these plots : Circle intensity = 0.7, wall intensity = 0.2.

output in response to a single valued visual array (ie the circle is not within the network's receptive field). Because of the balanced LoG filtering, the null activation is independent of the single value of the flat visual activation. The null response does however vary with the goal, because this changes the goal unit input to the network. In fig. 6.8, the animat will tend to move toward the circle at those places where the output in response to the circle is greater than the null response, since this implies that the output of the network facing the circle is greater than the output of the networks facing away. Where the output in fig. 6.8 is lower than the null response, output activation in the direction facing the circle will be lower than in directions facing away and so the animat will tend to move away from the circle. All of the directions in which the network does not view the circle will have equal, null responses, and hence are equally likely to be chosen as the movement direction. Thus the movement away from the circle toward the goal region is stochastic rather than direct.

The vertical lines in fig. 6.8 show the goal regions. For goal 1 (fig. 6.8a), the goal region is any distance greater than 128 from the circle. Output activation is lower than null activation for all distances smaller than the minimum goal distance. Hence, output activation is lower for the direction facing the circle, than for those facing away, and so the animat moves stochastically away from the circle as seen in fig. 6.7a.

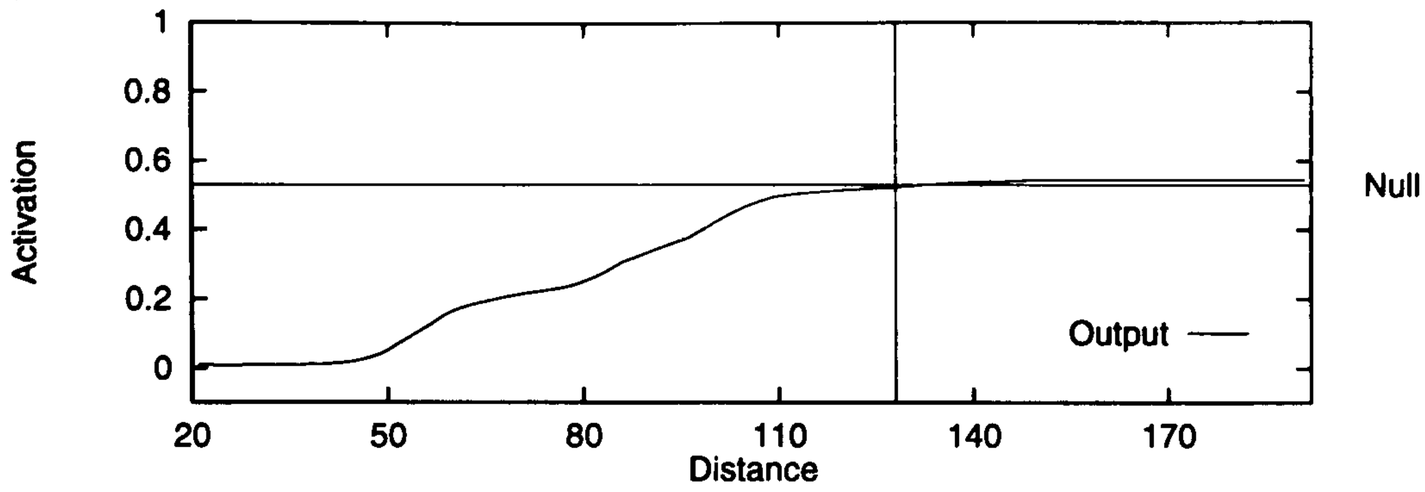
For goal 2 (fig. 6.8b), the goal region is distances between 77 and 128. At distances greater than this, output activation is greater than the null response, and hence the animat will tend to move toward the circle. Activation increases somewhat with decreasing distance to the goal region, reflecting the decreasing number of steps to positive reinforcement. At distances closer to the circle than the goal region (ie less than 77), output activation is much lower than the null response, and so the animat stochastically moves away from the circle, and hence toward the goal.

For goal 3 (fig. 6.8c), the goal region is distances between 56 and 77. The output activation profile is of similar form , but more sharply defined, than for goal 2. At distances further from the circle than the goal region, activation is greater than the null response. At distances closer to the goal region it is much less.

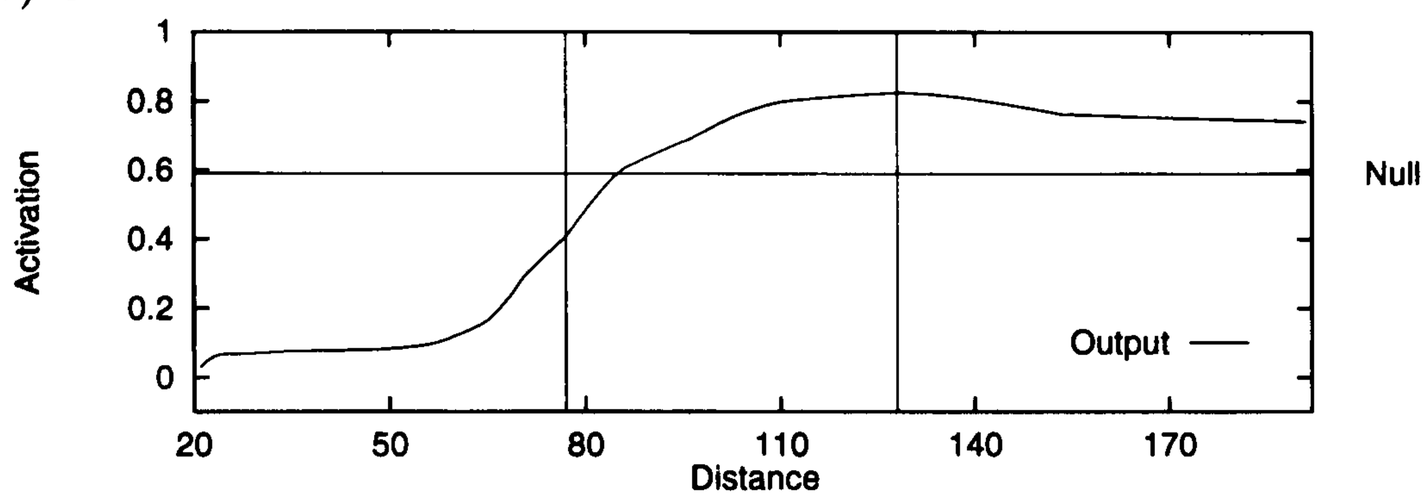
Goal 4 (fig. 6.8d), at distances between 44 and 56, elicits the sharpest output profile. The form, and hence the corresponding behaviour, is similar to goals 2 and 3.

Within the constraints of the stochastic aspects of animat control, the above output profiles entirely determine behaviour. The considerable differences between output profiles for the four goals are due to the response of the network's hidden units to the activation of the four goal units added to the input layer.

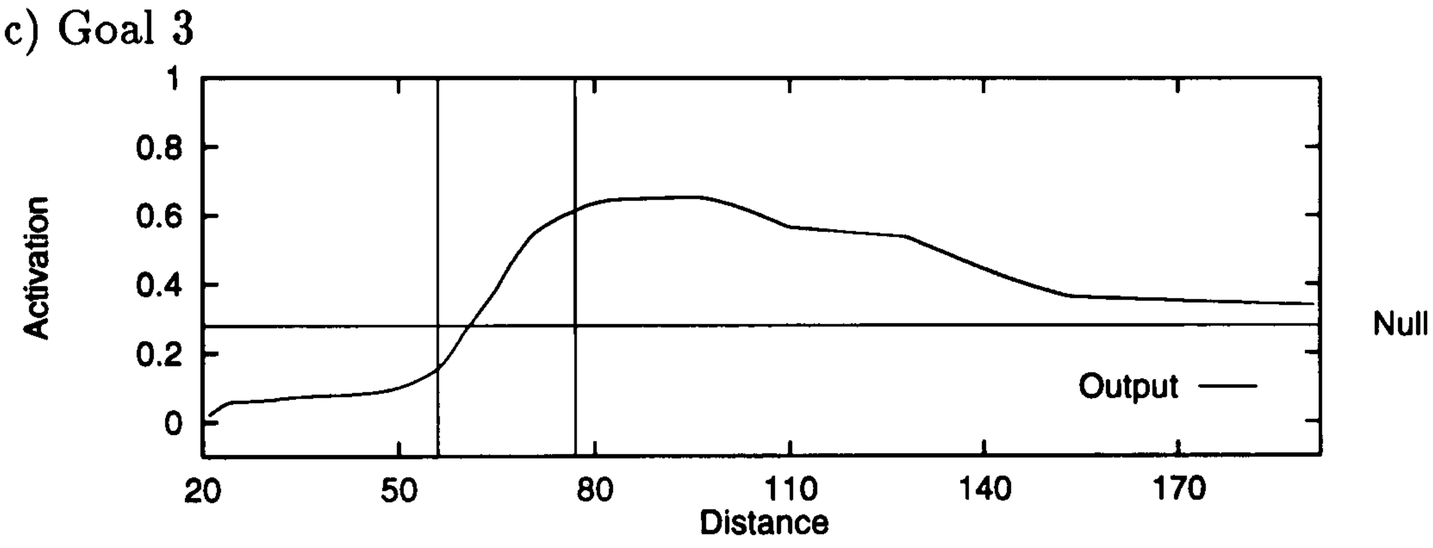
a) Goal 1



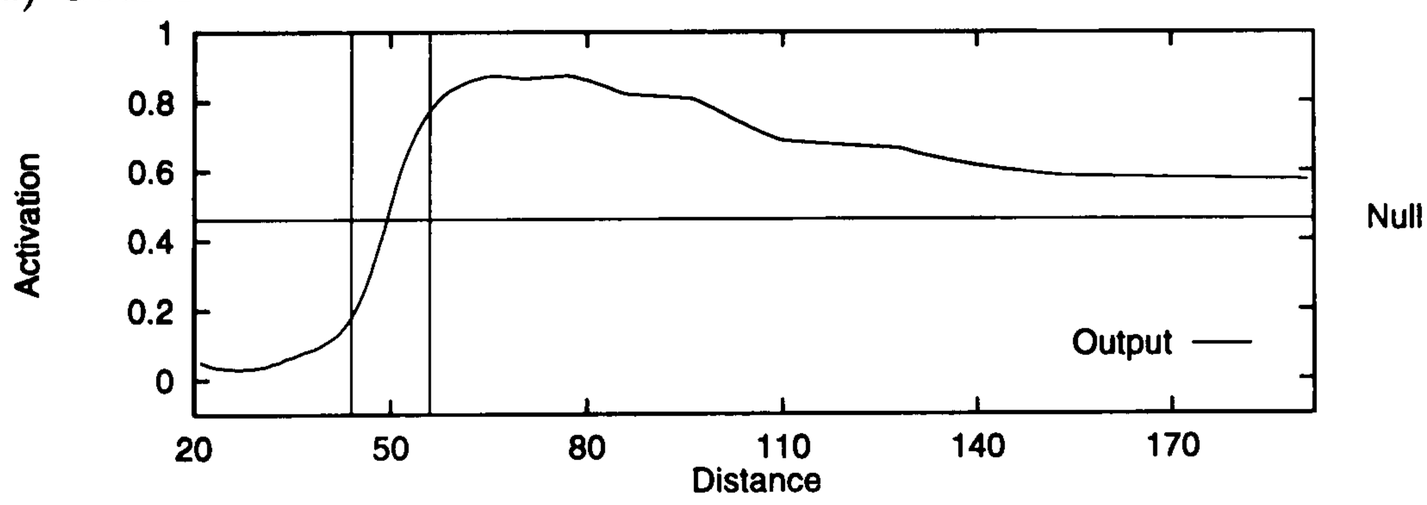
b) Goal 2



c) Goal 3



d) Goal 4



Output Unit Bias : 0.20.

Figure 6.8: The activation of the output unit of the filter network controlling the 4 hidden unit, rectified multiscale coding animat whose behaviour was shown in the previous figure. The activation is in response to the circle at the center of the filter networks receptive field, at the distance given by the horizontal axis. Activation is plotted for each of the four goal conditions. The horizontal line in each figure marks the unit's null response — the output unit activation when the circle is not within the network's receptive field. The vertical lines in each plot mark the distances between which the circle subtends the goal range of angles.
For these plots : Circle intensity = 0.7, wall intensity = 0.2.

6.6 Internal structure

The behaviour and output activation profile of a 4 hidden unit, rectified multiscale coding animat has been examined in the previous two sections. This animat has learned to perform well at the current task, and its behaviour can easily, and accurately, be inferred from the response of the output unit of the network controlling the animat. The next stage in analysis is to determine how the output function arises as a combination of the activation of the 4 hidden units. The two aspects of what hidden units have learned are the weight structure and the response profiles.

Figs. 6.9 and 6.10 show the learned weights of each of the hidden units of the animat whose behaviour and output unit response was examined above. Each hidden unit has weights to both the positive and negative multiscale filtered arrays. As seen in previous chapters, the weights to the positive and negative arrays tend to be very similar, reflecting the symmetry about zero contrast of the visual diet. In addition to the two arrays of weights to the sensory input, each hidden unit has weights to each of the 4 goal units. Only one of the goal units has activation 1 on each trial, the rest have activation 0. The effect on each hidden unit is to add the weight corresponding to the goal to the sensory input. Thus, these weights can be thought of as four separate, goal dependent, biases. Where the weight to a goal unit is positive, the activation of the hidden unit will be increased, regardless of sensory input, and where it is negative, the unit's activation will be decreased. The other two parameters associated with each hidden unit are its bias and weight to the output unit, also shown in the figures. The only learned parameter of this network not shown in figs. 6.9 and 6.10 is the bias of the output unit, which in this case is 0.20.

In addition to the learned network parameters, figs. 6.9 and 6.10 show the activation of each hidden unit in response to the circle placed at the center of the networks receptive field. The activation is plotted as a function of goal number and distance, in the same manner as for the output unit activation in fig. 6.8. In the following, the learned weight structures and activation profiles for each hidden unit in turn are examined.

Hidden unit 1 (fig. 6.9a), has large positive weights concentrated on the 2 coarsest scales, with large negative weights concentrated on the central region of the 3 finest scales. Hence, this unit will tend to be active when the circle subtends a large angle. Unit 1 has a large positive weight to the third goal unit, a large negative weight to the fourth goal unit and near zero weights to goal units 1 and 2. The unit has a negative bias, and so requires sensory plus goal input of greater than this to achieve an activation of greater than 0.5. The unit has an inhibitory connection to the output unit. The result of this unit's weight structure is that the unit has activation of 1.0 in goal condition 3, regardless of the circles distance; high activation when the circle is closer than 50 in goal condition 4; and similar activation profiles for goal conditions 1 and 2, of activation decreasing from 1.0 to 0.0 as the circle's distance increases from about 80 to 130. The response

in goal conditions 1 and 2 reflect the learned sensory weights, since in these conditions the input from goal units is smallest. The responses in goal conditions 3 and 4 are highly affected by the goal units.

Hidden unit 2 (fig. 6.9b), has large positive weights concentrated on the coarsest scale, with positive weights of lesser magnitude on the next from coarsest scale. Negative weights are concentrated in the central region of the three finest scales. With regard to the goal unit weights, unit 2 has negative weights to goal units 1 and 4, a positive weight to goal unit 2, and a near zero weight to goal unit 3. However, the magnitude of these goal unit weights are much less than for unit 1, whereas the sensory weights have about the same magnitude. Hence, the goal does not affect activation as strongly for this unit as it did for unit 1. Goal conditions 1 and 4 lead to identical responses, with activity decreasing from 1.0 to 0.0 as the circle's distance increases from about 40 to 60. In goal conditions 3 and 4, the activity falls away more slowly with increasing distance.

Hidden unit 3 (fig. 6.10), in contrast to hidden units 1 and 2, has positive weights at the finest spatial scales and negative weights at the coarsest scale. Weights get increasingly larger as the sale decreases, suggesting that this unit responds when the circle subtends a small angle, and hence is at a large distance. The weights of this hidden unit to the goal units are positive to goal unit 4, and negative to the other goal units. Looking at the activation plot, it can be seen that this unit does not respond to the circle at any distance, regardless of the goal. This is due to the large negative bias; sensory input cannot overcome this bias, and so the unit does not become active. Hidden unit 3 has a large inhibitory weight to the output unit, but seemingly never becomes active enough to influence it. The unit must have had an effect during learning at some stage, or else the weights would not have become structured, as they clearly are in fig. 6.10. Over the course of learning, the influence of the unit must have gradually been diminished by an increasingly negative bias, until, as seen here, the unit has no influence at all. Presumably the unit became active when the circle was at a far distance. Since, as seen in the previous section, the output unit responds usefully in these conditions, one of the other hidden units must have taken over the function of this hidden unit.

An intriguing finding is that the other two simulations of animats in this condition (rectified multiscale with 4 hidden units), also have a hidden unit with a similar weight structure to hidden unit 3 in this network. Furthermore, these units, like the one in this network, have such strongly negative biases that their response is small or zero in all conditions. This supports the above argument that such units may function usefully during early stages of learning, but become faded out of the final network function. A more detailed analysis of the formation, and development over time, of hidden unit weight structure could be undertaken to investigate these issues. Here however, the consistency of the weight structure, and its puzzling lack of involvement after extended learning

are merely noted.

Hidden unit 4 (fig. 6.10), has positive weight concentrated on the second and third from coarsest scales. Goal unit weights are negative for goal 1, positive for goals 2 and 3, and near zero for goal 4. Unlike the other three hidden units, hidden unit 4 has an excitatory weight to the output unit. The unit has a zero response in goal condition 1, for all distances. For goals 4,3 and 2, activity peaks at around distance 50, and decreases with increasing distance. Because of the positive goal unit input, activity decreases less rapidly for goal numbers 2 and 3.

The weighted combination of the activation functions plotted in figs. 6.9 and 6.10 determine the output activation function shown in fig. 6.8. The output unit function, in turn, stochastically determines the behaviour shown in fig. 6.7.

6.7 Discussion

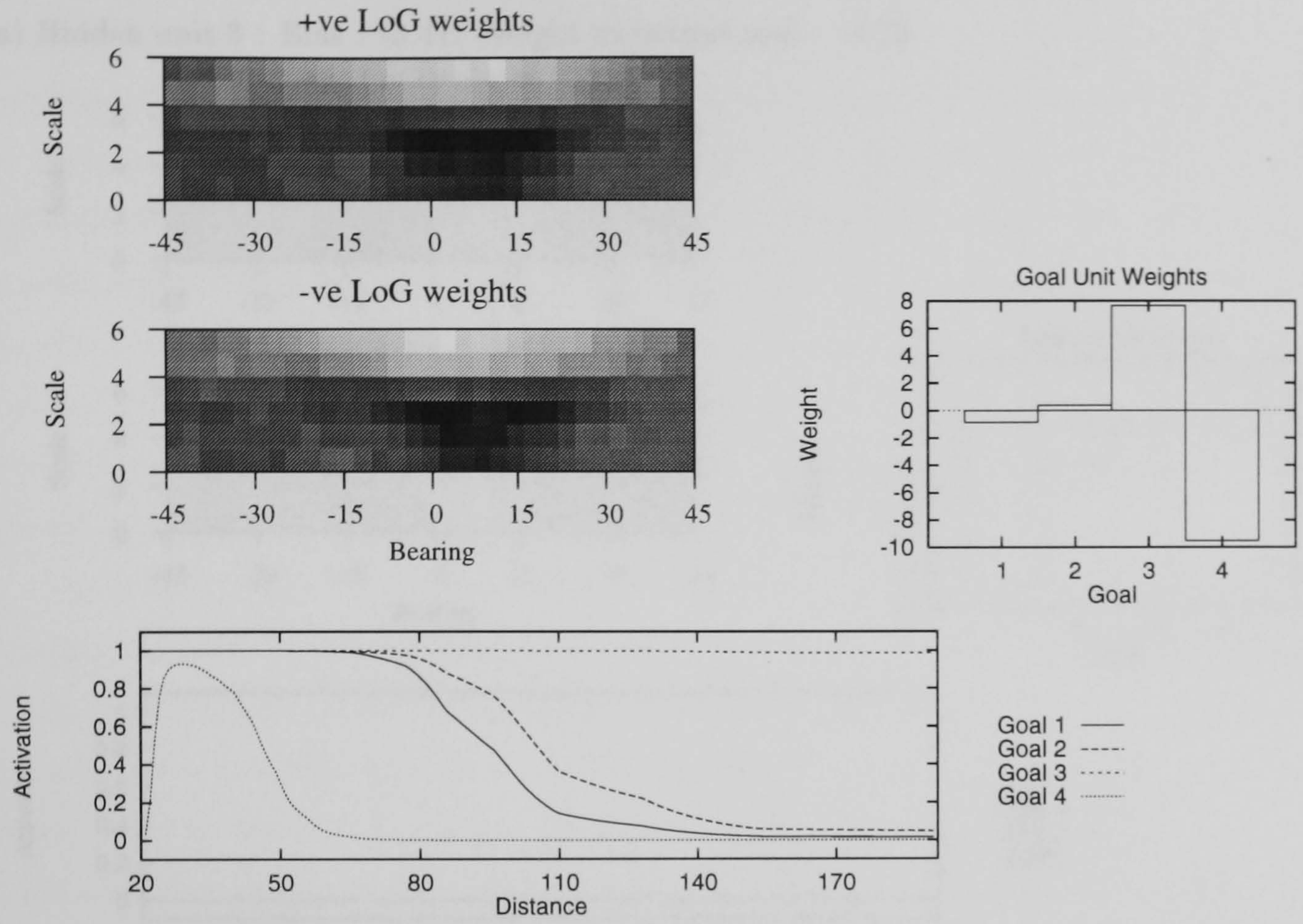
The task studied in this chapter is a more complex extension of that of the previous chapters. In the previous two chapters, the network controlling the animats had to learn a single mapping from the visual array to a particular output. Each location within the arena demanded a unique behaviour, either to move toward, or away from the circle, and the correct behaviour was determined solely by the visual array. In this chapter, the task demands that animats learn behaviours depending upon both the visual array, and the activation of auxiliary units coding the current goal.

Animats using intensity coding were found to be unable to learn this multiple subtended angle task, even though no sensory noise is present. Rectified multiscale coding animats, in contrast, with 2 or more hidden units perform well at the task, with some increase in performance with more hidden units. Examination of the behaviour of these animats shows that they have learned to both approach, and move away from the circle, in the correct situations. The difference between intensity coding and rectified multiscale coding in this task is the largest seen so far in this thesis without noise. Given that this chapter's task differs from previous ones in terms of complexity, it can be suggested that the advantage of rectified multiscale coding becomes more apparent as the visual task becomes more complex.

When tested on the goal range corresponding to the task of the previous chapter, rectified multiscale animats were shown to achieve a level of performance matching that of animats that had learned this single goal range. Good performance over all four goals ranges can be learned by animats with 4 hidden units, whereas it required 2 hidden units to learn the single goal range of the previous chapter. This suggests that animats have learned to decompose the stimuli set in a computationally more economical manner than would be expected from the results with a single goal range.

A rectified multiscale coding animat with 4 hidden units has been examined in detail. The

a) Hidden unit 1 : Bias : -2.40. Weight to output unit : -1.51.



b) Hidden unit 2 : Bias : -2.88. Weight to output unit : -3.46.

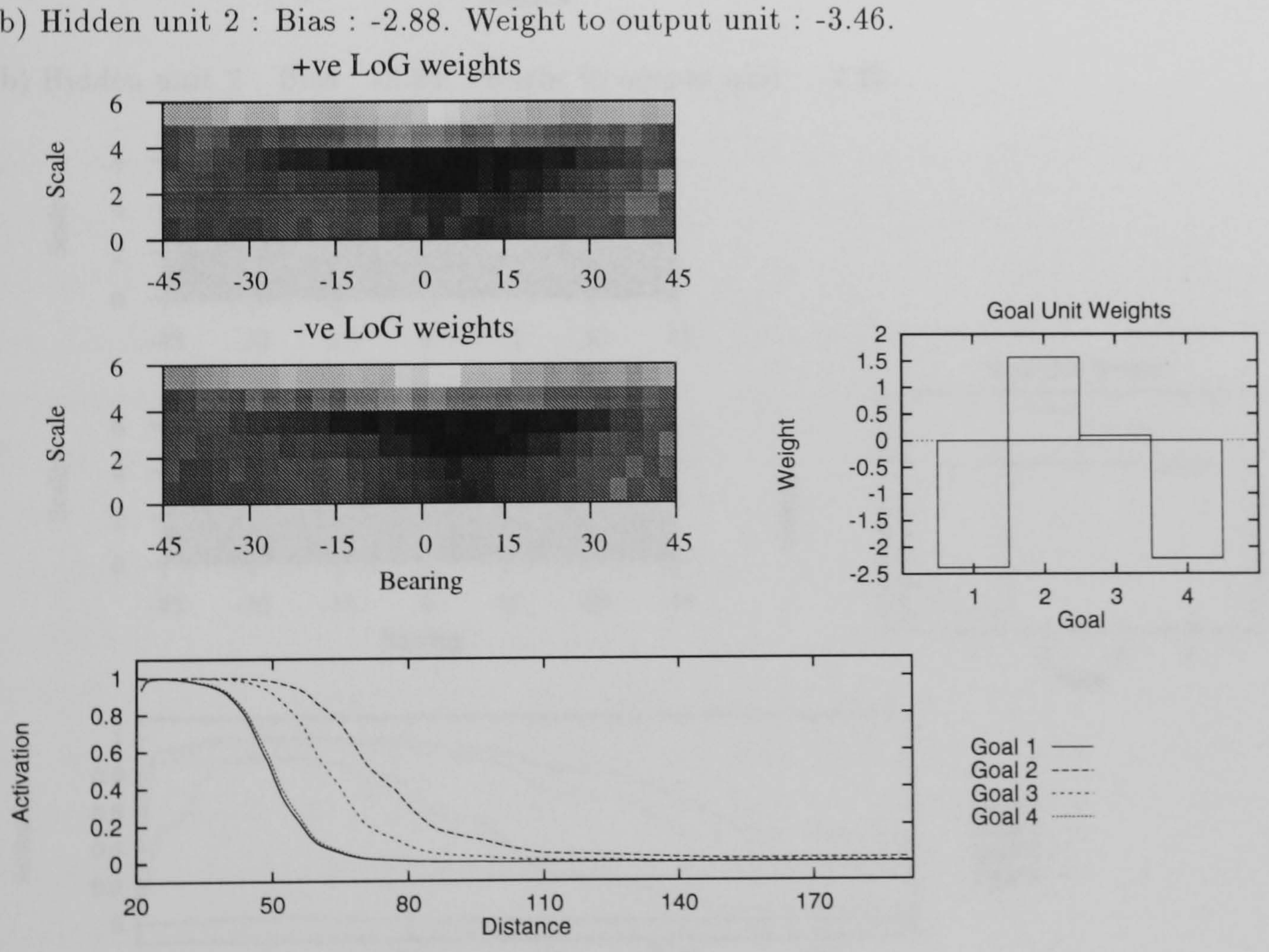
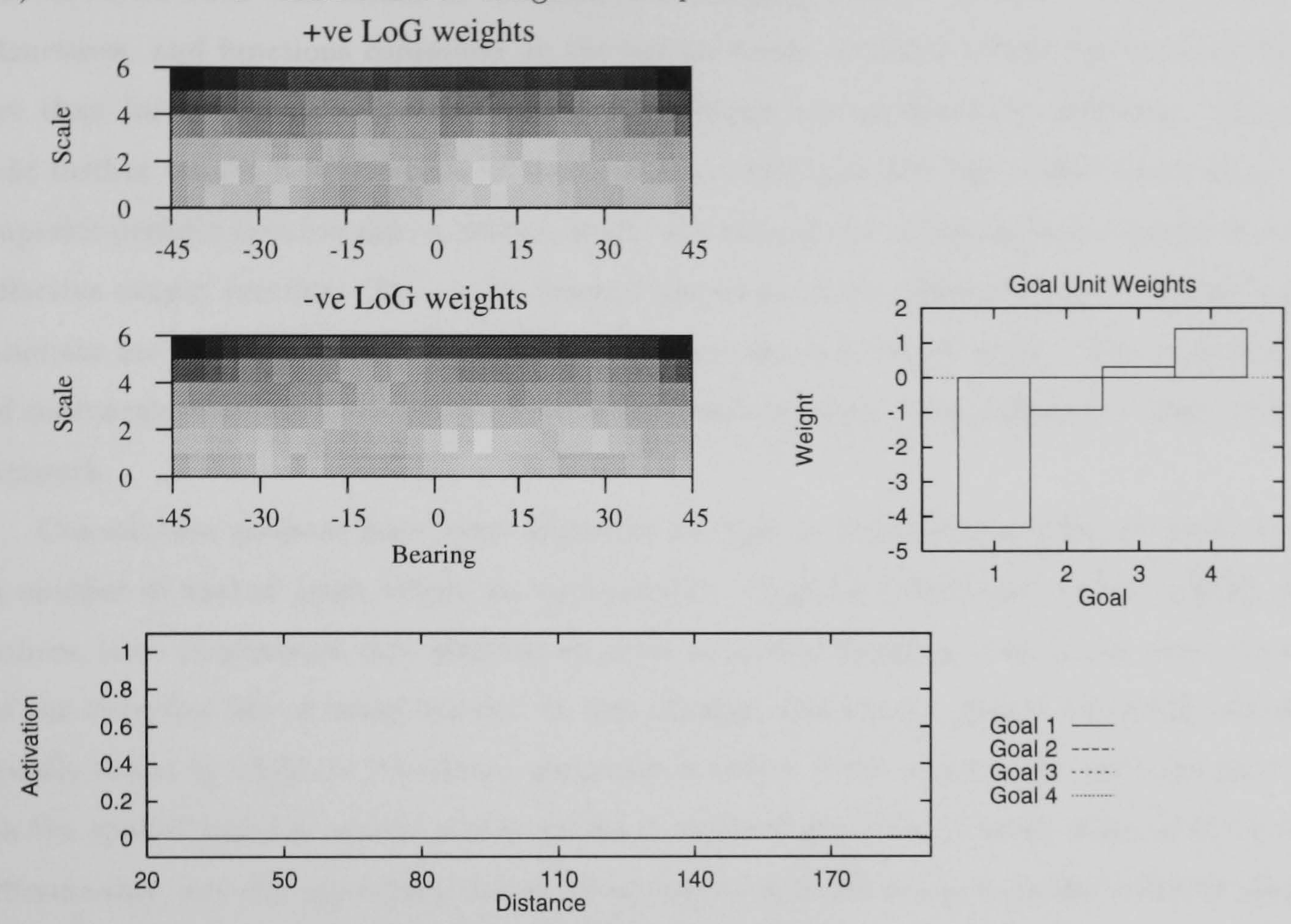


Figure 6.9: The learned weight structure and response profile of hidden units 1 and 2 of the 4 hidden unit, rectified multiscale coding filter network controlling the animat whose behaviour and output unit response profile were shown in the previous two figures. The first row shows the learned weights; the second row shows the activation of this unit in response to the circle for each of the four goal conditions. (continued in next figure)

a) Hidden unit 3 : Bias : -3.47. Weight to output unit : -4.25.



b) Hidden unit 2 : Bias : -2.24. Weight to output unit : -2.23.

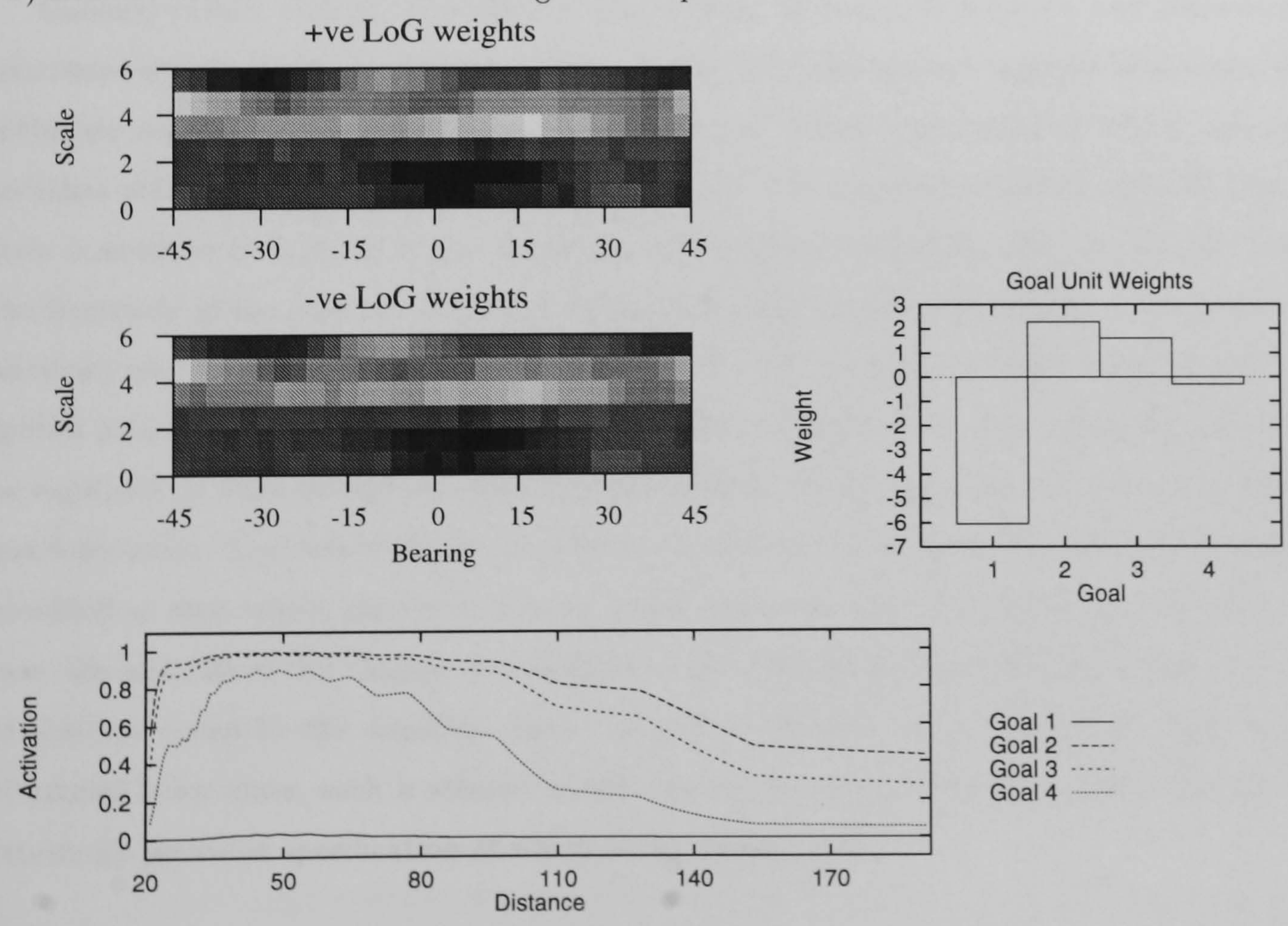


Figure 6.10: (Continued from previous figure) Hidden units 3 and 4. Details as per previous figure

relationship between the output unit activation profile, and the behaviour of the animat, was shown to be both well suited to the task and comprehensible. Analysis of the learned weight structures, and functions computed by the hidden units, revealed a finer partition of the stimuli set than for the single subtended angle task, though not qualitatively different. These results add further weight to the argument that rectified multiscale filtering of the visual array leads to superior performance because it makes simpler the subsequent computations required to map to an effective output function. The subtle learned responses of the hidden units of rectified multiscale animats are just too complex to be learned from the raw visual array. The expanded coding of multiscale filtering does much of the computational work, thus taking the load off the filter network.

Convolution animats have been shown to be able to concurrently learn to move to each of a number of spatial goals within an environment. Gallistel (1990) and Collett (1992), amongst others, have emphasised that learning to move to several foraging sites at the same time is part of the everyday life of many insects. In this chapter, the current goal is externally decided, and locally coded by which of the binary goal units is active. Over many trials, animats learn to move to the spatial location where reinforcement is received given the current state of the goal units. Presumably, any distinguishable set of binary patterns could act to code the different goals. This suggests interesting extensions of the work in this chapter.

Gallistel (1990) reviews the evidence that insects, including honeybees, can learn to move to particular spatial goals at particular times. Gallistel (1990) further suggests that many animals, including insects, code time by means of a string of binary units each of which automatically switches state with a different temporal frequency. The smallest temporal interval that such a code is sensitive to is given by the frequency of the fastest switching unit; the longest interval by the frequency of the slowest. All times within this range are uniquely coded. If the externally set, auxiliary goal units of this chapter were replaced by such an automatically changing code, and the spatial goal where reinforcement is received varied regularly with time, then the animats could be expected to learn to move to different spatial goals depending upon the time, as coded by the auxiliary units. This would clearly happen in the simplest case where the task of this chapter was modified so that which goal unit was on varied automatically and regularly with time. In this case, the animats of this chapter would move to the different subtended angle ranges as a function of time, as coded by the auxiliary units. As well as offering the possibility of modeling a range of animal behaviours, such a scheme would remove the computationally unsatisfactory need for externally provided specification of which is the current goal.

Chapter 7

Reinforcement Landmark Learning

7.1 Introduction

7.1.1 Gerbil Landmark Learning

Collett, Cartwright and Smith (1986) trained gerbils to find a food reward at a fixed location relative to an arrangement of identical cylindrical landmarks. The environment for these experiments was a circular arena painted black, and lit by a single bulb, illuminating a central region and leaving the walls in darkness. The floor was covered in wood chips and as far as possible, all sensory cues to the food location were removed, except for the visual stimuli provided by an arrangement of white cylinders acting as landmarks. The food location itself was invisible. Between trials, the array of landmarks and relative food location was translated, but not rotated, and gerbils were released from random start locations within the arena. This ensures that the animals learn paths relative only to the landmarks and not other visual cues.

Within about 150 trials, gerbils had learned to run to the food location from any start location. Once the gerbils were trained to criteria, they were tested by occasional probe trials in which food was absent. A histogram of search frequency at each location was obtained to determine where the animal searches during these trials. This is assumed to be a measure of where it expects the food to be. More importantly, test trials were conducted with the arrangement of landmarks modified from that during learning. The distribution of search effort in the modified environments provides crucial evidence about the computations underlying the animals behaviour by enabling models of the behaviour to be separated. Candidate models will learn to search in the goal location when tested with the environment the same as during learning. However, when tested

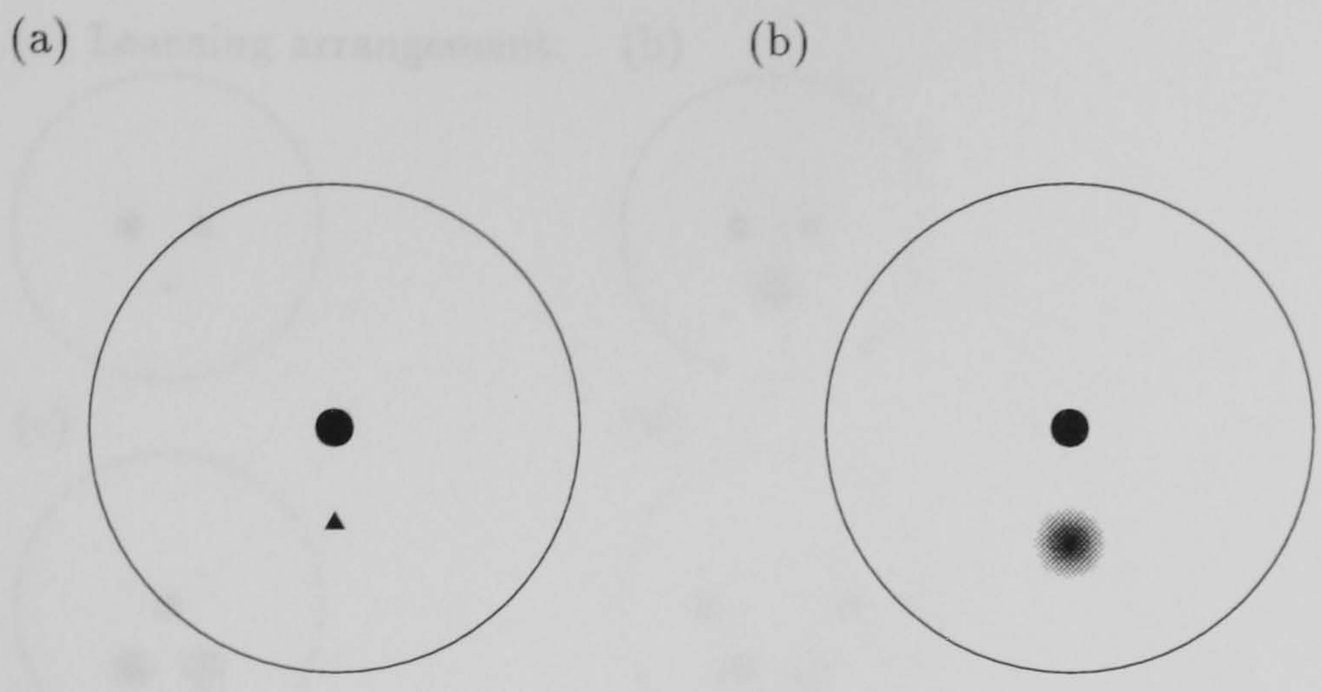


Figure 7.1: Task 1. (Highly schematic from Collett et al (1986)) (a) Learning situation : The cylindrical landmark is shown as a black circle and the (invisible) food source as a triangle. (b) 2 dimensional gray scale search histogram, with higher search frequency locations darker. The dark patch shows where gerbils, after learning, spent most time searching on the test trials where no food was present.

with modified landmark arrangements, models that differ significantly will tend to predict different search locations. The predictions can then be compared with the gerbils behaviour and the model rejected if they do not match.

7.1.2 Task 1

In the simplest environment used by Collett et al (1986), the arena contained a solitary cylinder and food was located at a fixed distance and direction from this single landmark (fig. 7.1a). Animals learn to move to the food location upon release from the start box, as shown by their search histogram (fig. 7.1b). That the gerbils can learn a bearing from a featureless cylinder implies the use of a non-visual direction sense, most probably either from a magnetic compass sense (evidence for mammals reviewed by Gallistel, 1990, Baker, 1980; shown in insects by Collett and Baron, 1994), or from integration of head rotation as specified by the vestibular system (McNaugton et al, 1995).

Collett et al (1986) manipulated the radius of the cylinder in order to determine what visual cues are used to compute distance. Cartwright and Collett (1983) did the same experiment with bees and found that the visual angle subtended by the image of the cylinder specified how far away they searched. Thus, when tested with a cylinder of twice the training radius, bees searched at twice the distance. With a cylinder half the training radius, they searched at half the distance. With gerbils the situation is not so clear; with a landmark half the training size, gerbils searched at half the distance as predicted by the subtended angle hypothesis. However, when tested with a cylinder bigger than in training, their search position was unaffected.

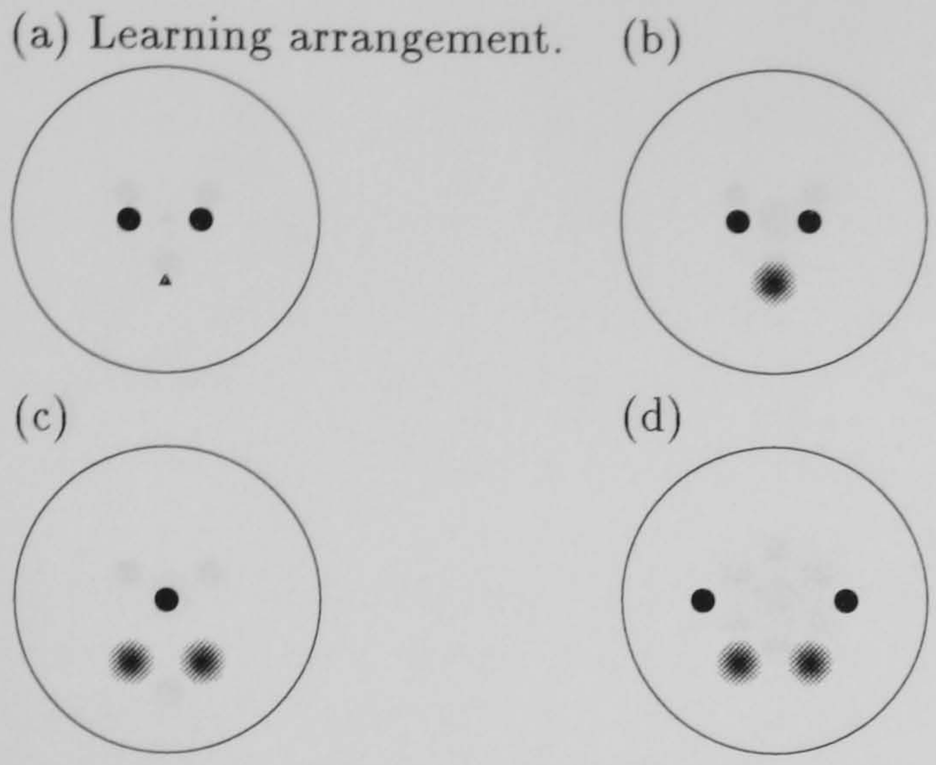


Figure 7.2: Task 2. (Highly schematic from Collett et al (1986)) (a) Learning situation : The 2 cylindrical landmarks are shown as black circle's and the (invisible) food source as a triangle. (b)–(d) Search histograms. The dark patches show where gerbils, after learning, spent most time searching on the test trials where no food was present. (b) Search when tested with the same array as during learning. (c) Search when tested with a single landmark. (d) Search when the distance between the landmarks is doubled.

7.1.3 Task 2

With the goal and two landmarks arranged as shown in fig. 7.2a, gerbils learnt the food location as shown by their search pattern when tested with the training arrangement of landmarks (fig. 7.2b).

When tested with only one cylinder, gerbils spent most time searching in the two locations shown in fig. 7.2c. Note that this is the same environment as in fig. 7.1; differences in prior learning cause the very different search behaviours. Here, each of the two search locations is at the same bearing and distance from one of the landmarks as was the goal during learning. This, and the results for task 1, lead Collett et al (1986) to suggest that gerbils learn to move to the location where the bearing and distance of individual landmarks matches their bearing and distance from the goal. However, when tested the distance between the landmarks is doubled, gerbils searched in just two of the four locations suggested by this hypothesis (fig. 7.2d).

7.1.4 Task 3

Here, the food is located at the center of an equilateral triangle as shown in fig. 7.3a. Gerbils learn to go to the goal location as shown in fig. 7.3b.

When tested with the distance of one of the landmarks from the center doubled, gerbils still searched in the same location as the goal during learning (fig. 7.3d). This suggests that when two out of three landmarks are in the normal position, gerbils ignore the outlier. When tested with a single landmark, gerbils searched at three locations, each of which is at the same bearing and distance with respect to one of the landmarks as was the goal during learning (fig. 7.3c). Testing with the triangle of landmarks rotated leads to the search behaviour shown in fig. 7.3e.

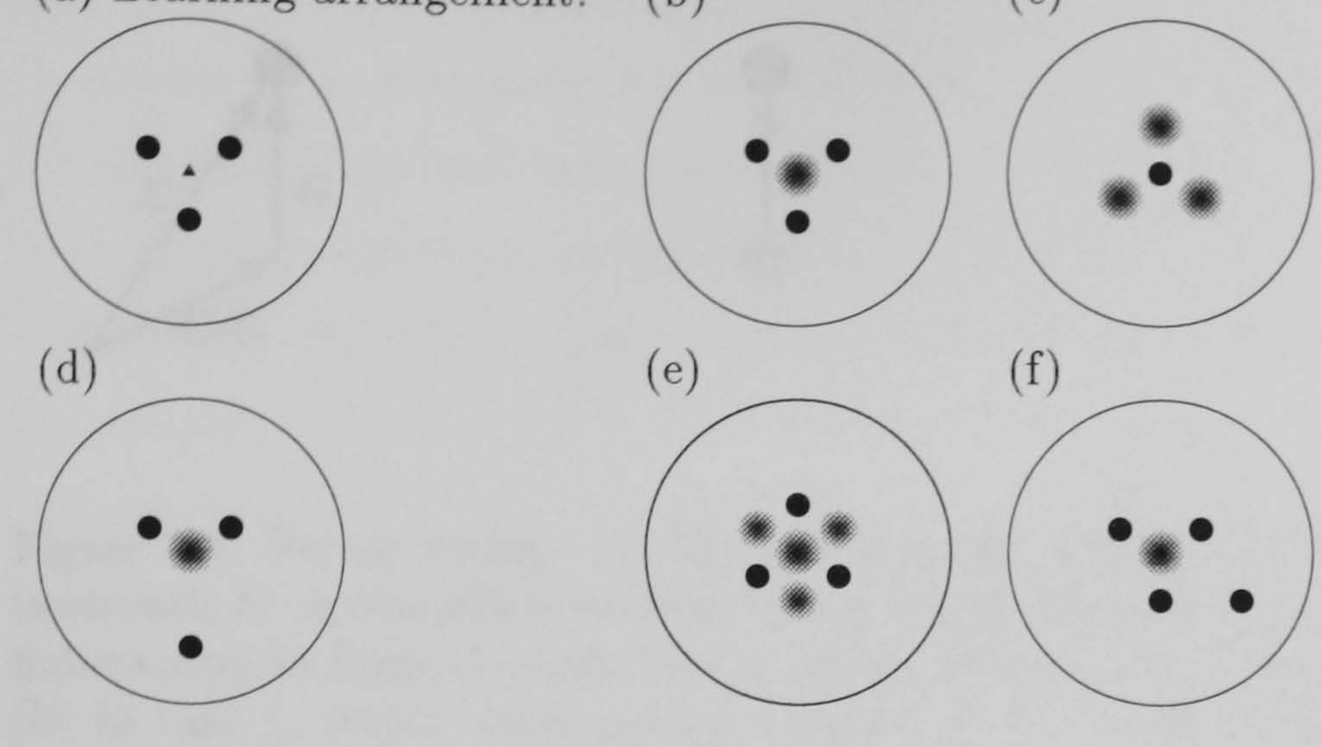


Figure 7.3: Task 3. (Highly schematic from Collett et al (1986)) (a) Learning situation : The 3 cylindrical landmark are shown as black circle's and the (invisible) food source as a triangle. (b)–(e) Search histograms. The dark patches show where gerbils, after learning, spent most time searching on the test trials where no food was present. (b) Search when tested with the same array as during learning. (c) Search when the distance between 1 of the landmarks and the rest is doubled. (d) Search when tested with a single landmark. (e) Search when the landmark array is rotated through 60 degrees. (f) Search with extra landmark.

The gerbils direct most of their search to the center of the rotated array, suggesting a knowledge of the relation between the three landmarks. In addition, they searched to a lesser extent in three locations outside of the landmark triangle. Each of these locations is at the same distance and bearing to two of the landmarks as was the goal in the training array. In fig. 7.3f, an extra landmark gives the gerbils a choice between landmarks in the training arrangement and their rotation. In this case, the animals search where the arrangement matches that experienced during learning.

Collett et al (1986) tested the gerbils behaviour when the light was turned off a short time after release and found that the animals were still able to search in the goal location. Furthermore, the pattern of search in tests with the modified landmark arrangements remained largely the same as that shown when the light was on. This indicates that the gerbils plan trajectories to the goal rather than reactively responding to their current visual input.

7.2 Models of the gerbil's behaviour

7.2.1 Vector voting

Collett et al (1986) propose vector voting, a simple model of what gerbils learn, which makes concrete and falsifiable predictions of where they will search after learning with a particular landmark arrangement. Vector voting captures much of the search behaviour described above, but as emphasised by Collett et al (1986), it is limited in that some of its predictions with multiple land-

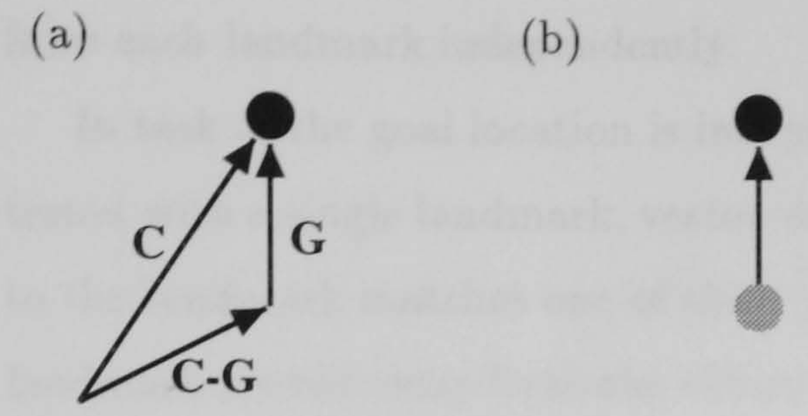


Figure 7.4: Vector voting: (a) At each location, the vector linking the current location to each landmark, **C**, is computed, and the vector linking the goal location to each landmark, **G**, is stored. Subtracting **G** from **C** results in the vector linking the current location to the goal location. (b) In task 1, vector voting predicts search at the location (shown in gray) where the vector to the landmark is the same as during learning.

marks do not replicate gerbil search behaviour. Vector voting forms the basis of current models of the search behaviour, including Touretzky and Redish (1995), Redish and Touretzky (1997) and McNaughton et al (1995), which are, in behavioural terms, implementations and elaborations of the basic computational hypothesis.

Vector voting assumes that the gerbil uses allocentric direction information to compute the distance and allocentric bearing of each landmark independently, resulting in a vector linking the current location to each landmark. At the goal location, the set of vectors, one for each landmark, linking the goal location to each landmark is stored. As shown in fig. 7.4a, at any location, subtracting the goal vector from the current landmark vector results in the vector linking the current location to the goal location. This specifies the trajectory for movement to the goal. In task 1, vector voting predicts search at the correct distance and direction from the single landmark, and thus matches the gerbil's behaviour (fig. 7.4b).

With multiple landmarks, each landmark independently specifies search locations, and so in tests with rearranged landmarks, multiple goal locations will be specified. When each location is specified by the same number of landmarks, gerbils are assumed to search in each. When some locations are specified by more landmarks than others, the gerbil is assumed to search at the location consistent with the largest number of landmarks.

In task 2, two landmarks each specify the goal and so two goal vectors of distance and allocentric bearing are stored (fig. ??a). When tested with a single landmark, two locations match the goal distance and allocentric direction and so vector voting predicts search at both of these, which replicates the gerbil behaviour (fig. ??b). However, when the distance between the two arrays is doubled, vector voting specifies search in four locations (fig. ??c), whereas gerbils only search at the inner two locations. Since the landmarks are identical, the gerbils behaviour implies that in addition to the independent landmark vectors, they can also use the relation between landmarks to guide movement. This is not possible with vector voting because trajectories are computed

from each landmark independently.

In task 3, the goal location is independently specified by 3 vectors (fig. 7.5a). Therefore, when tested with a single landmark, vector voting predicts search at the three locations where the vector to the landmark matches one of these (fig. 7.5b). This replicates the gerbils behaviour. With one landmark moved away from the others, eight locations have the stored vector to a landmark, but one location has two vectors supporting it, and so vector voting predicts search there (fig. 7.5c). Again this replicates the gerbils behaviour. With the rotated triangle of landmarks, three locations have 2 vector votes, as shown in fig. 7.5d; gerbils search in these locations but also search at the center of the rotated triangle. This is not predicted by vector voting, since at the center of the rotated triangle, none of the landmarks have the same allocentric bearing as during learning. As Collett et al (1986) point out, only sensitivity to the relative bearing between landmarks would lead to search in the center of the rotated triangle. With an extra landmark (fig. 7.5e), vector voting predicts search at the center of the unrotated triangle, since this receives three votes as during learning. This matches the gerbils behaviour.

Of the gerbil search patterns with modified landmark arrangements outlined in section 7.1.1, vector voting correctly predicts all but two: the split landmarks in task 2 (fig. 7.2d), and the rotated landmarks in task 3 (fig. 7.3e). Collett et al (1986) suggest that these failures are due to the assumption that landmarks are treated independently, and the model is sensitive only to allocentric rather than relative bearings. Gerbils appear to be sensitive to both the allocentric bearing of individual landmarks, and the relative bearings between multiple landmarks. Collett et al (1986) further supported this by showing that gerbils can learn a goal location specified by multiple landmarks that are rotated, as well as translated, between trials.

7.2.2 Current models of rodent navigation

Touretzky, Redish and Wan, develop a model of rodent navigation, simultaneously modeling both animal behaviour, and neurophysiological data on hippocampal place cells and cells sensitive to head direction (Reddish and Touretzky, 1997; Touretzky and Redish, 1995; Touretzky, Wan, and Redish, 1994; Wan, Touretzky, and Redish, 1994a, 1994b). The model specifies the interaction between simulated place units, head direction units, and a path integrator, and predicts search patterns in the Collett et al (1986) tasks as well as a number of other experiments.

Input to the model is a list of the distance and egocentric bearing of each currently visible landmark, together with the change in head direction, used to update the model's current estimate of head direction. Allocentric bearings of the landmarks are computed from egocentric bearing and head direction, and the path integrator, an estimate of the current Cartesian position with respect to a reference point, is updated from the new estimate of head direction (Touretzky and Redish, 1995). Unlike vector voting, in which the trajectory is computed as a simple function of

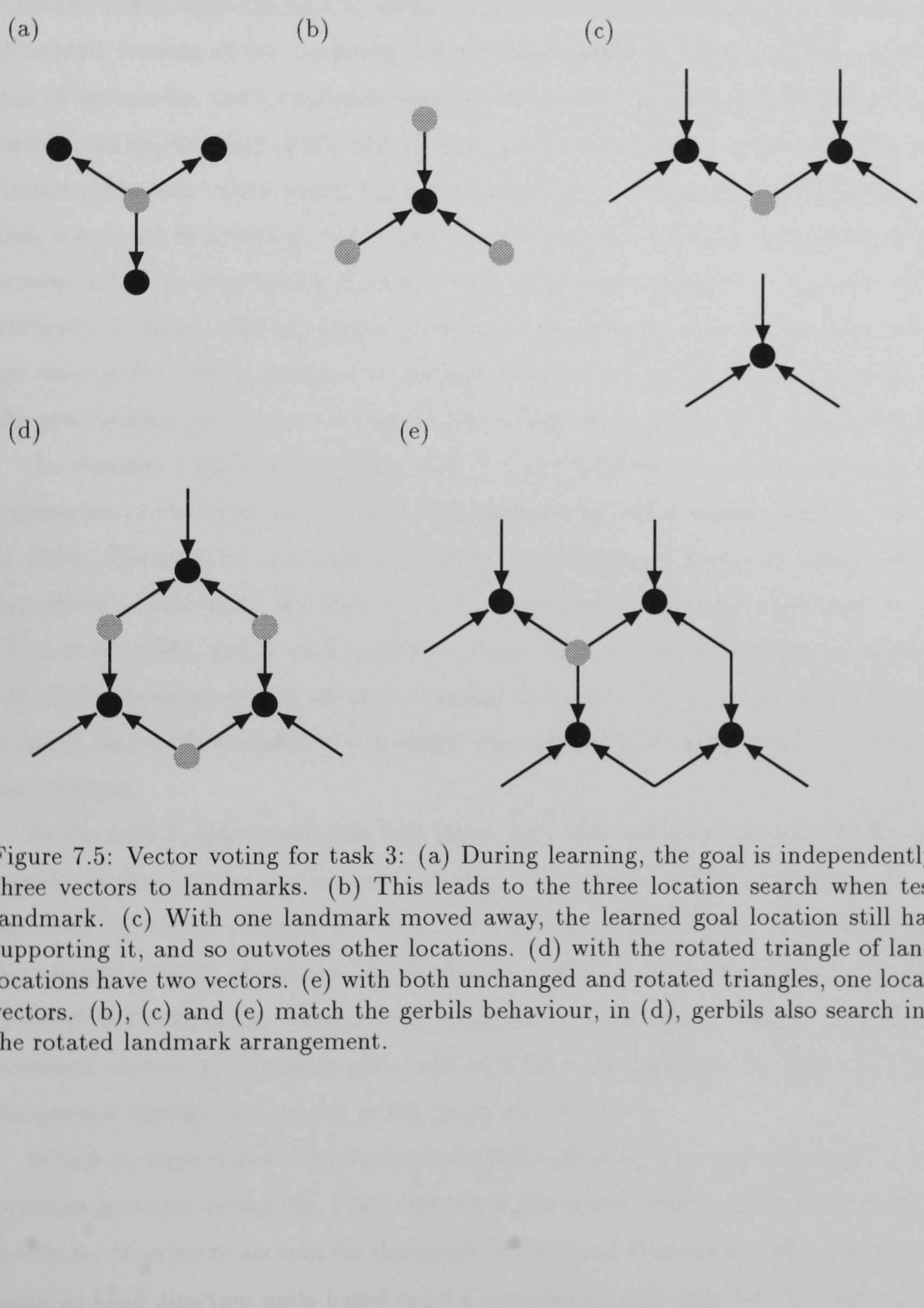


Figure 7.5: Vector voting for task 3: (a) During learning, the goal is independently specified by three vectors to landmarks. (b) This leads to the three location search when tested with one landmark. (c) With one landmark moved away, the learned goal location still has two vectors supporting it, and so outvotes other locations. (d) with the rotated triangle of landmarks, three locations have two vectors. (e) with both unchanged and rotated triangles, one location has three vectors. (b), (c) and (e) match the gerbils behaviour, in (d), gerbils also search in the center of the rotated landmark arrangement.

the input, Touretzky et al's model requires a large number of place units, each of which coarse codes for a particular Cartesian allocentric location within the environment. The system uses the place units to estimate where it is in allocentric coordinates, and then, like vector voting, uses explicit vector subtraction to compute a trajectory to the goal location.

Learning in the model involves place units learning the correlation between current landmark cues and the path integrator's current estimate of allocentric position. Each place unit is a Gaussian radial basis function broadly tuned to the independent dimensions of the distance and allocentric bearing of two randomly selected landmarks, the angle between a randomly selected pair of landmarks, and a Cartesian location as specified by the path integration system. During learning, at any location in the environment, place units become active according to how well the current parameter values match the stored values; if no place units become sufficiently activated, then a new one is recruited. Over time, the environment becomes covered with place units that become active in overlapping locations, each of which is sensitive to a subset of the cues at a particular location. Moving in the environment is guided by first determining which place units are most active, which specifies the current estimate of location. Subtracting this location from the goal location, as in vector voting, results in the vector linking the current location to the goal.

To simulate Collett et al (1986) task 2, this model requires 3,000 place units to cover the environment (Touretzky and Redish, 1995); simulating task 3 requires 10,000 place units (Wan et al, 1994). The authors argue that their model is "a distributed implementation of the vector voting hypothesis" (Touretzky and Redish, 1995), "producing behaviour equivalent to vector voting" (Wan et al, 1994), and as such, mostly predicts the same search patterns as vector voting in the test trials. As shown above, vector voting fails to account for two gerbil search behaviours in tasks 2 and 3, and so Tourestzky et al's model can only replicate these results by elaborating on the basic system.

In the task 2, split landmarks test (fig. 7.2d), Touretzky and Redish (1995) show that their model, like the gerbils, search mostly at the two locations within the landmarks out of the four predicted by vector voting (fig. ??b). This result occurs because place units are sensitive to the relative bearing of pairs of landmarks, in addition to the distance and allocentric bearing of individual landmarks. However, the comparative weighting of relative bearing and individual landmark vectors to determine place unit activity is an externally set parameter and hence so is the systems likelihood to search at the outer two locations.

In task 3, when tested with the rotated landmark array, the system searches in the same three locations as vector voting (fig. 7.5c), but not in the center of the rotated triangle of landmarks like gerbils do. In order to account for this result, Redish and Toutetzky (1997) propose that the model resets its head direction units based upon a complicated and explicitly specified comparison of the difference between each stored allocentric landmark bearing and its current egocentric bearing. A

consistent difference results in resetting of the head direction units and hence the reference bearing from which allocentric bearings are computed.

McNaughton et al (1995) develop a model of rodent navigation, that, like Touretzky et al's, models both behavioural and neurophysiological data. In this system, head direction is used to compute a vector coding of landmark distance and allocentric bearing. This is then subtracted from the vector linking the goal location to the landmark to provide the trajectory, as in vector voting and Touretzky and Redish's (1995) model. Vector operations are implemented in McNaughton et al's (1995) model by feedforward neural networks. The model is an implementation of Collett et al's (1986) vector voting, and as such, predicts search in the same locations, and fails to account for the same gerbil test results as vector voting.

7.3 Simulations

7.3.1 Sensory coding

In the previous chapters of this thesis, sensory input consisted of a visual array of intensity values. This was processed by the adaptive convolution network either raw or after convolution with multiscale filters. Given the impoverished visual environment of featureless cylinders within a featureless arena, and the reactiveness of animat processing, subtended angle is the only cue in the visual array (or multiscale filtered array) available to the animats to judge distance. Chapter 5 showed that animats can learn to use visual subtended angle to guide search around a solitary circle, and hence search at different distances when tested with circles of different radius. In this the animats behaviour matches that of honeybees (Collett and Cartwright, 1983). Gerbils however do not exclusively use the visual angle subtended by the cylinder to guide search, since replacement of the cylinder with one of twice the radius does not affect search location.

It would therefore it be pointless to use the same visual array and reactive filtering and processing as in the previous chapters to try and model the gerbils behaviour. Francesini et al (1992) show how convolution by filters with a simple spatiotemporal structure yields distance information. In the simulations of this chapter, a computation such as this is assumed in a schematic form and sensory input to animats consists of an array of landmark distances.

Animats have a 360 degree view provided by 60, evenly spaced receptor units with non-overlapping receptive fields, each coding the distance of any landmark falling within its receptive field. Landmarks only cause activity in the single receptor whose bearing they are closest to, and the activity of all other receptors is set to zero.

Two methods of coding distance are compared through simulation: intensity coding, in which a single, continuous valued unit at each bearing codes distance, and vector coding, in which a

number of binary units at each bearing code distance between them.

Landmark distance in each direction is first converted to a receptor activity between 0 and 1 by the function $f(d) = 2\sin^{-1}(10/d)$, capped at 1. This function is the angle in radians subtended by a circle of fixed radius 10, it is monotonic with distance and falls exponentially from 1 to 0 as distance increases, as shown in fig. 7.6a.

Intensity Coding

With intensity coding, a single continuous valued unit for each direction codes the receptor activity, resulting in a 1-D sensory array. In these simulations, the 1-D intensity coded sensory array has 60 elements (fig. 7.6c, left column).

Vector coding

With vector coding, instead of having a single, continuous valued unit coding distance in each direction, a number of binary units are used, each coding for a non-overlapping range of receptor activations. Hence, the 1-D receptor array is expanded to a 2-D array. At each landmark distance, a single one of these units will be on and the rest off. A vector code is an array in which the relative position of a unit, as well as its activation, carries information (Gallistel (1990)). In a vector code, each element in the array can be regarded as a vector, and the whole array as an ordered set of vectors. Binary units provide the most extreme example of a vector code because information is carried entirely by which unit is on rather than by unit activation levels. McNaugton et al (1995) propose a vector coding of landmark distance and allocentric bearing. Here, the vector coding is of distance and egocentric bearing.

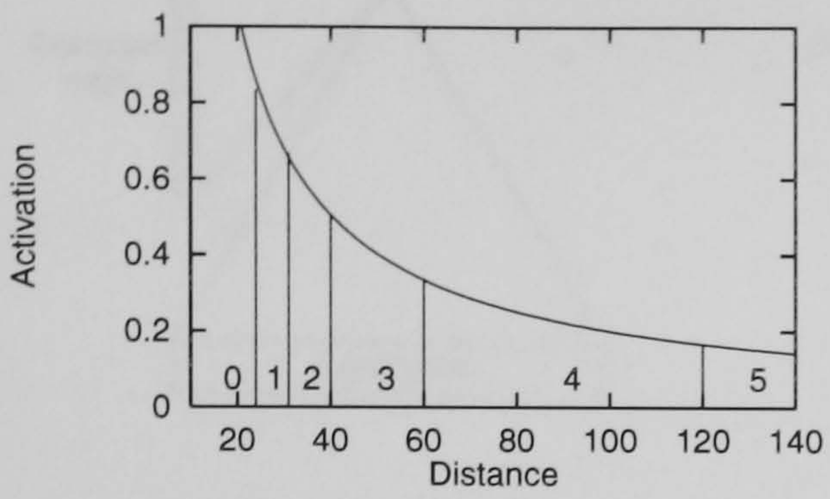
In these simulations, receptor activity (between 0 and 1) is split into 6 equally sized regions, corresponding to increasingly large distance ranges. Hence the 2-D vector coded sensory array is of size 6×60 (fig. 7.6c, left column). Because only six binary units are used, vector coding of distance is a great deal coarser than the continuous valued intensity coding.

7.3.2 Animat processing

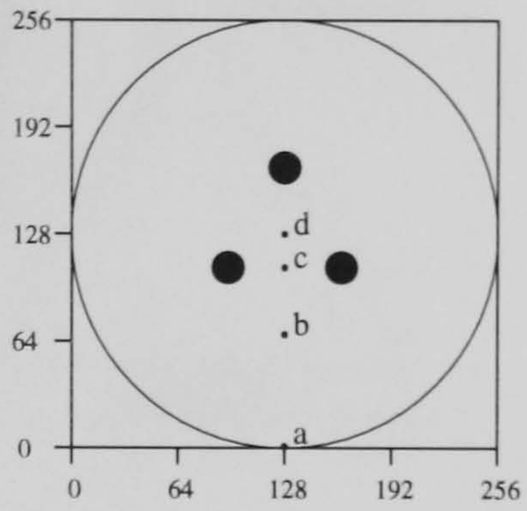
The intensity, or vector coded array is convolved with a filter network in the same way as in the previous chapters to produce a 15 element, 1-D motor array. The motor array stochastically determines the direction in which the animat moves a fixed distance; the higher the value of a motor array element, the higher the probability that the animat will move in the corresponding direction. Filter networks are standard feedforward networks with a single output unit, and either a single layer of weights (direct) or hidden units and two layers of weights (fig. 7.7).

In addition to the sensory input, filter networks also have a layer of auxiliary input units to

(a)



(b)



(c)

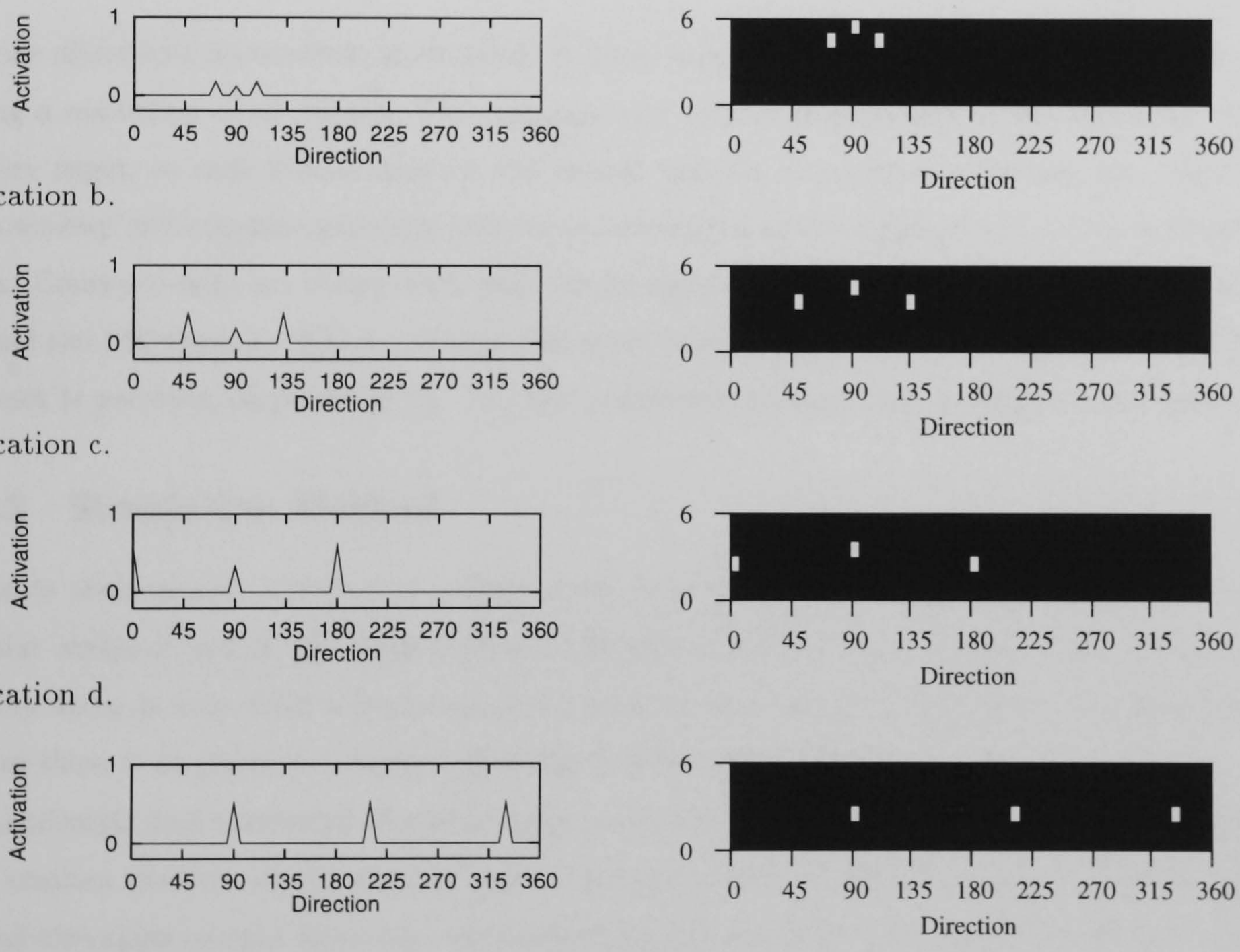


Figure 7.6: **Sensory Coding.** (a) Landmark distance in each direction is converted to a receptor activity between 0 and 1. With intensity coding, this value is coded by a single unit in each direction. With vector coding, each receptor activity is coded by 6 binary units with non-overlapping ranges. The vertical lines plot the boundaries between these ranges.
(c) plots the 1-D intensity coded sensory array (left column) and the 2-D vector coded array (right column) for each location a–d in (b).

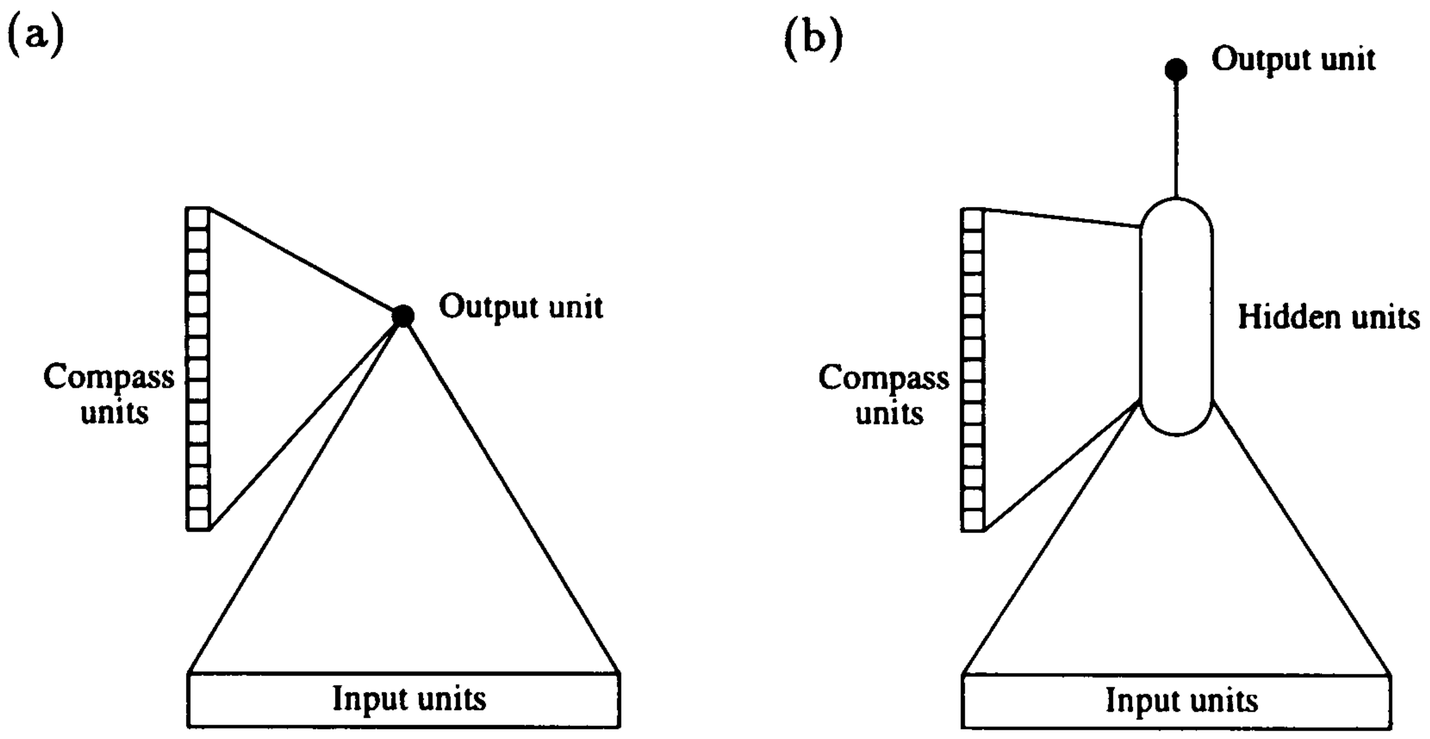


Figure 7.7: Filter networks: (a) direct, (b) with hidden units. In addition to the sensory input, hidden units also have weighted connections to binary compass units. In the case of direct filter networks, the output unit is connected to the compass units. The input layer is 1-D in the intensity coding condition and 2-D in the vector coding condition.

provide allocentric orientation information. In these simulations, 15 binary compass units are used giving a resolution of 24 degrees. The compass unit values are processed in the same way as the sensory input, so each hidden unit (or the output unit for direct filter networks) has weights to both sensory and compass units and activity is determined as the weighted sum across both sets of units. Compass units are binary with only one on and the rest off for each filter network position around the 360 degrees. Which compass unit is on depends upon the direction in which the filter network is pointing, as shown in fig. 7.8, with a different compass unit coding for each direction.

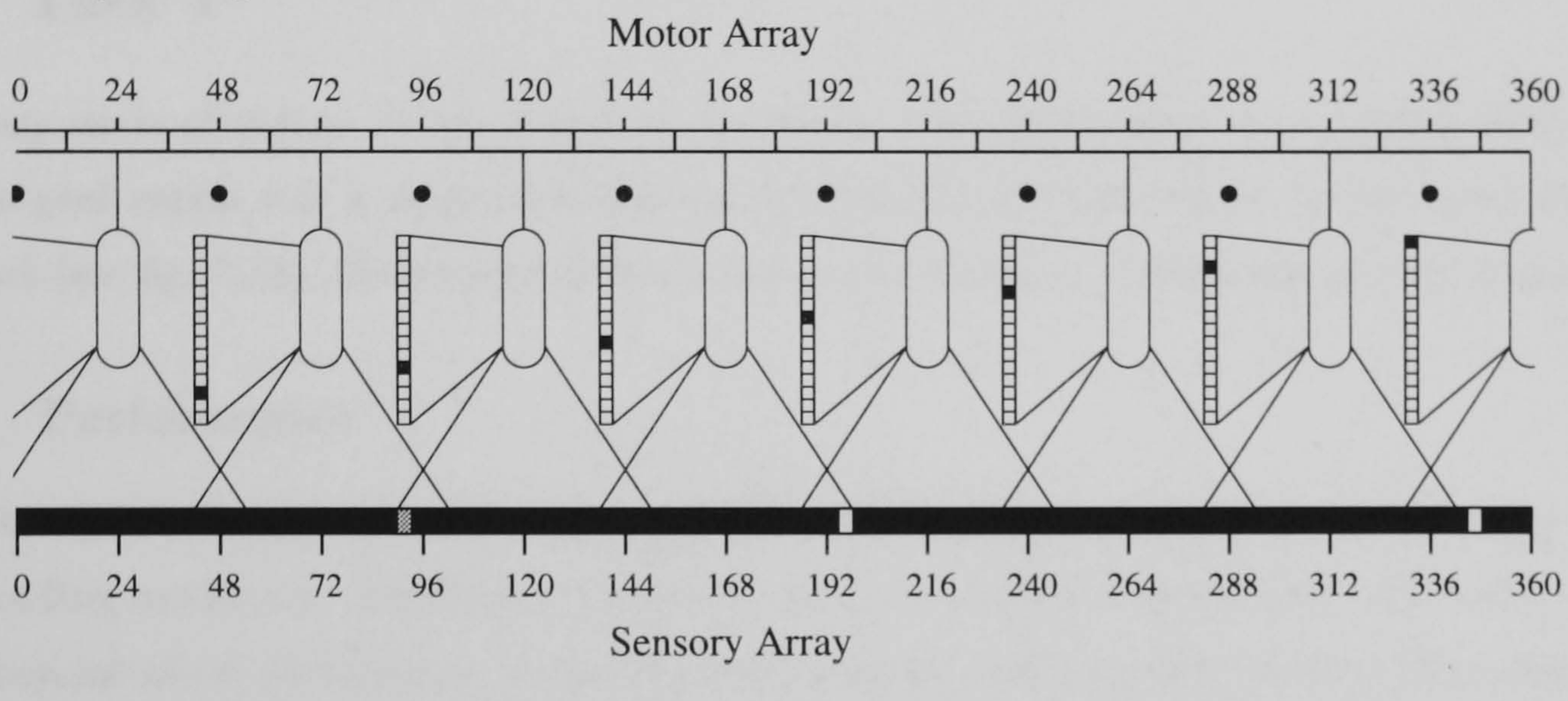
7.3.3 Simulation Method

Animats with various numbers of hidden units were simulated for the three tasks above, in a circular arena of radius 128 spatial units. Animats could not move outside this. Because the sensory array is zero when a landmark is not present, they are unable to detect the arena walls, and so there is no point in translating the landmarks between trials.

Landmarks had a radius of 10 spatial units, as did the circular goal region. Animats were placed at a random location at the start of each trial, and moved 10 spatial units in one of 15 evenly spaced directions on each time step, until either they landed in the goal region. Reinforcement was zero on all time steps, except when the animat moved into the goal region, when reinforcement of 1.0 was received, and a new trial begun. If animats did not get to the goal within the maximum time of 500 steps a new trial was begun.

All simulations were replicated three times with different random initial weights and random numbers. Reinforcement learning followed exactly the same algorithm as in previous chapters and outlined in chapter 3.

(a) Intensity coding



(b) Vector coding

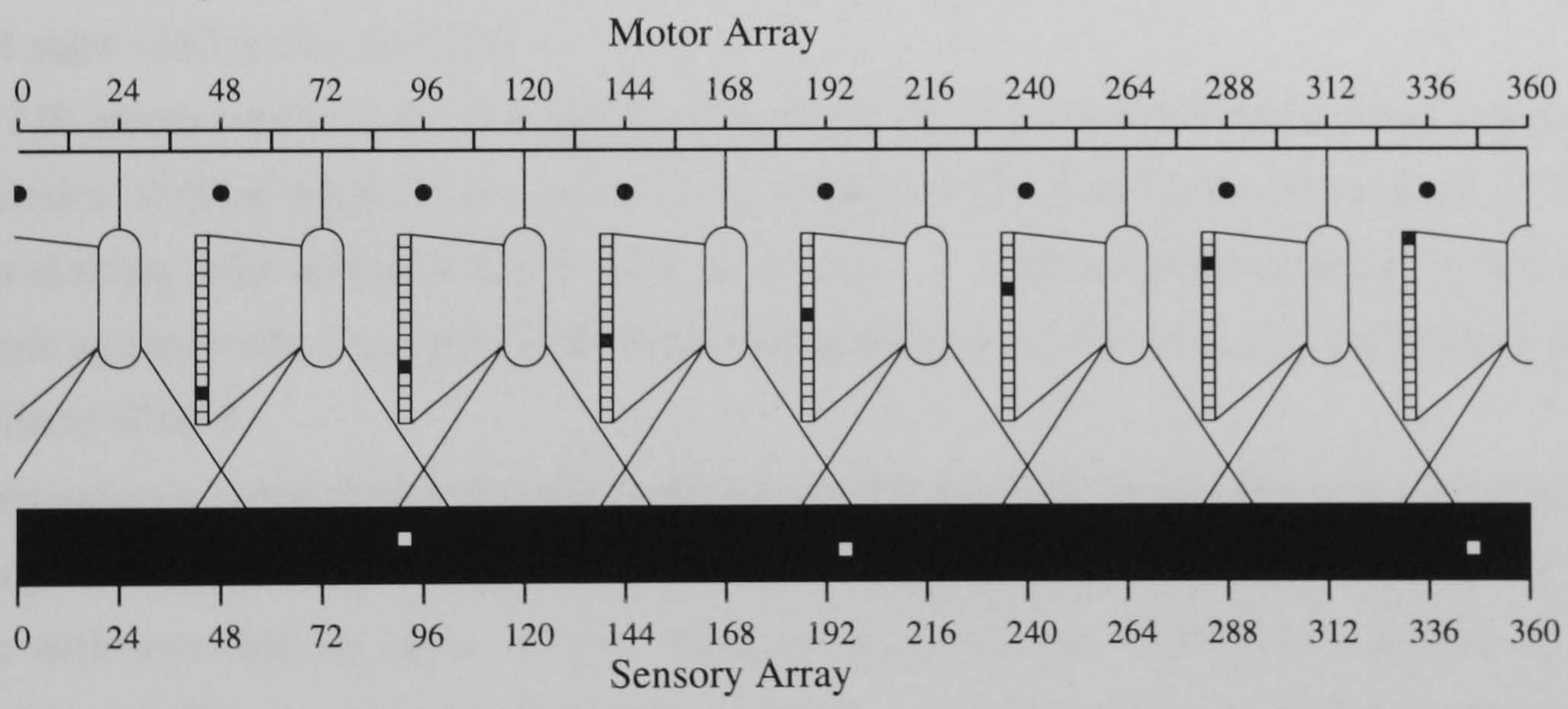


Figure 7.8: Animat sensorimotor system. The filter network is convolved (with subsampling) with the sensory array to produce a 15 element motor array. The sensory and motor arrays covers 360 degrees. A different binary compass unit is on for each direction in the convolution (dots mark undrawn filter network positions)

All animats were tested over 1000 trials without learning, with randomly chosen starting locations on each trial. Performance is measured as the minimum path length divided by the actual path length. Thus, performance is between 0 and 1.0; with 1.0 corresponding to an animat taking the shortest route to the goal. A performance of 0.5 means that the animat took, on average, twice as many steps to get to the goal as the shortest route.

7.4 Task 1

A solitary circle of radius 10 was placed at the center of the simulated arena (radius 128); the invisible goal region was a circle of radius 10, located at a distance of 30 to the south of the landmark (see fig. 7.10). Filter network fan-in was set at 31 units, corresponding to 93 degrees.

7.4.1 Performance

Fig. 7.9a shows learning curves for three animats with four hidden units, in both intensity and vector coding conditions. Performance improves until an asymptote is reached, well within 50K trials, beyond which performance remains about constant. Animats with vector coding learn in less trials and achieve higher performance than intensity coding animats, a pattern repeated with two and eight hidden unit animats.

Fig 7.9b shows performance after 50k learning trials, by which time all animats had converged, as a function of filter network size and sensory coding. Each data point is the mean of three animats starting with different initial random weights. A random animat, tested in the same conditions as those after learning (1000 trials with maximum of 500 steps per trial) has a mean performance of 0.06

Direct animats (without hidden units), perform little better than the random animat, regardless of coding. Intensity coding animats with 2 hidden units perform at around 0.4, and this does not increase with more hidden units. Vector coding animats with two hidden units perform at 0.5; this increases further to around 0.66 with four hidden units, but shows no further increase with eight hidden units. Four hidden unit, vector coding animats significantly outperform four hidden unit intensity coding animats (t-test : $t=6.426$, $p<0.01$).

7.4.2 Behaviour

Intensity coding

Fig. 7.10 shows the behaviour of a 4 hidden unit animat in the intensity coding case; this animat performs at 0.33. In fig. 7.10a, 20 example trials are shown, with the animat starting from equally spaced locations around the arena edge. The animat stochastically drifts south when north of

the landmark, moves more directly toward the goal bearing when somewhat to the south of the landmark, but has particular difficulty when located due south of the circle and goal (co-ordinates of about (128,0)).

Fig. 7.10b shows the animat's behaviour as a function of location. To generate this plot, the motor vector, which determines the probability of movement in each direction, was computed for each location in the environment. The largest of these probabilities, and hence the direction in which the animat is most likely to move is plotted as an arrow in fig. 7.10b. The darker the arrow, the higher probability that the animat will move in the direction of the arrow. The top half of fig. 7.10b has light arrows, generally pointing downward. This indicates that the animat has a somewhat higher than chance probability of moving downward, and leads to the stochastic drift south seen in fig. 7.10a. When the animat is located south of the circle (y coordinate of less than 128), the response is stronger near the goal, but the arrows point quite widely along a line to the south of the circle, and are not focussed upon the goal region. At the extreme south (y coordinate of near zero), the response is highly confused, leading to the random behaviour in this region seen in fig. 7.10a.

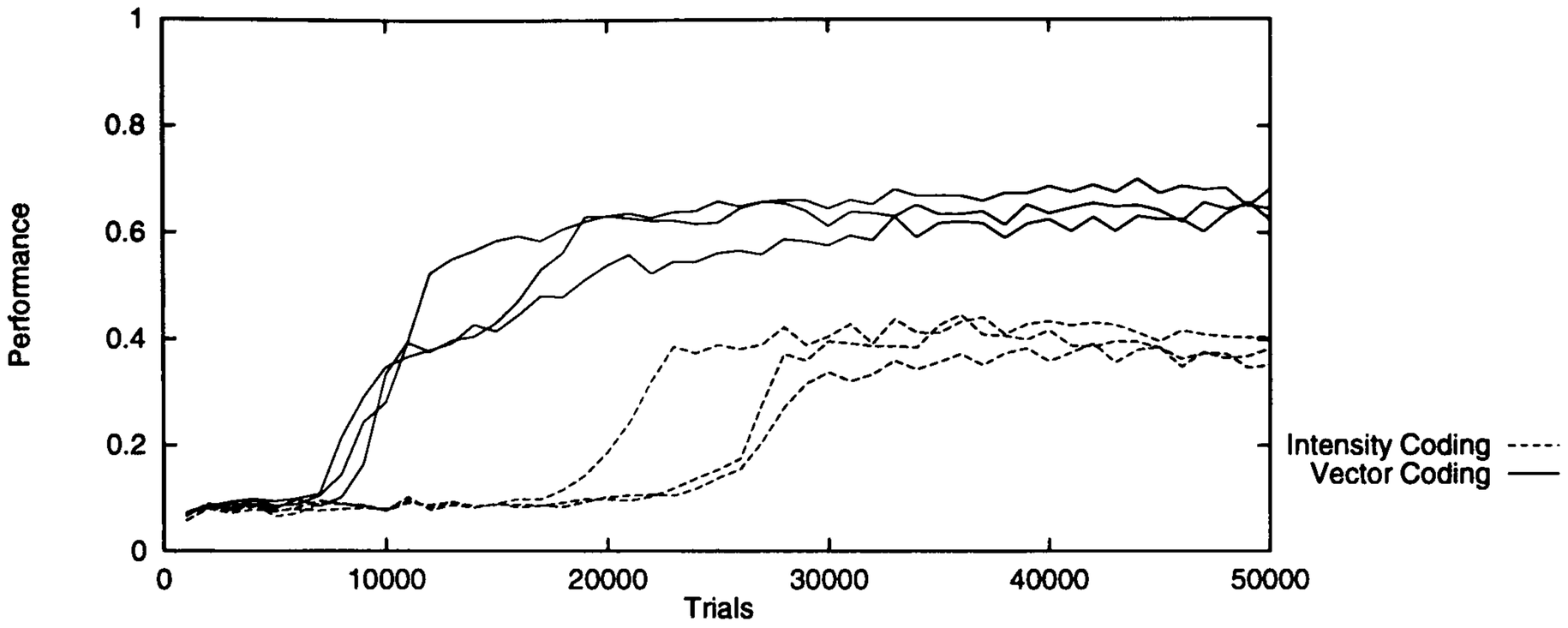
Fig. 7.10c shows the search histogram for this animat, in an analogous form to the search histograms obtained from animals by Collett et al (1986), and schematised in figs. 7.1–7.3. To generate the histograms, animats are placed in random locations within a goal-less arena and move for fifty time steps. This is repeated 100 times. Fig. 7.10c plots the proportion of time spent in each location, which reflects where the animat searches, and hence where the animat supposes the goal to be. The four hidden unit, intensity coding animat under analysis here spends most of its time searching close to the circle at roughly the correct bearing.

These results suggest that this intensity coding animat, whilst able to learn to move to where the bearing of landmark matches the bearing from the goal, is unable to move to the goal distance. Similar considerations consistently lead to the same conclusion for intensity coding animats on task 1.

Vector coding

Fig. 7.11 shows the behaviour of a vector coding animat with four hidden units. This animat performs at 0.69. The example paths of fig. 7.11a show that this animat moves stochastically southward when north of the circle, and fairly directly toward the goal when south of the circle. Fig. 7.11b) makes this behaviour clearer: north of the circle, the arrows are light and mostly point southward. South of the circle the arrows are darker, reflecting the stronger output in these locations and hence the greater probability of moving in the arrow's direction. In addition to having more accurate directions to the goal than the intensity coding animat of fig. 7.10, the vector coding animat has higher probability of moving in its preferred directions (reflected in the

(a) Learning Curves of Animats with 4 Hidden Units on task 1.



(b) Performance as a function of network size for task 1.

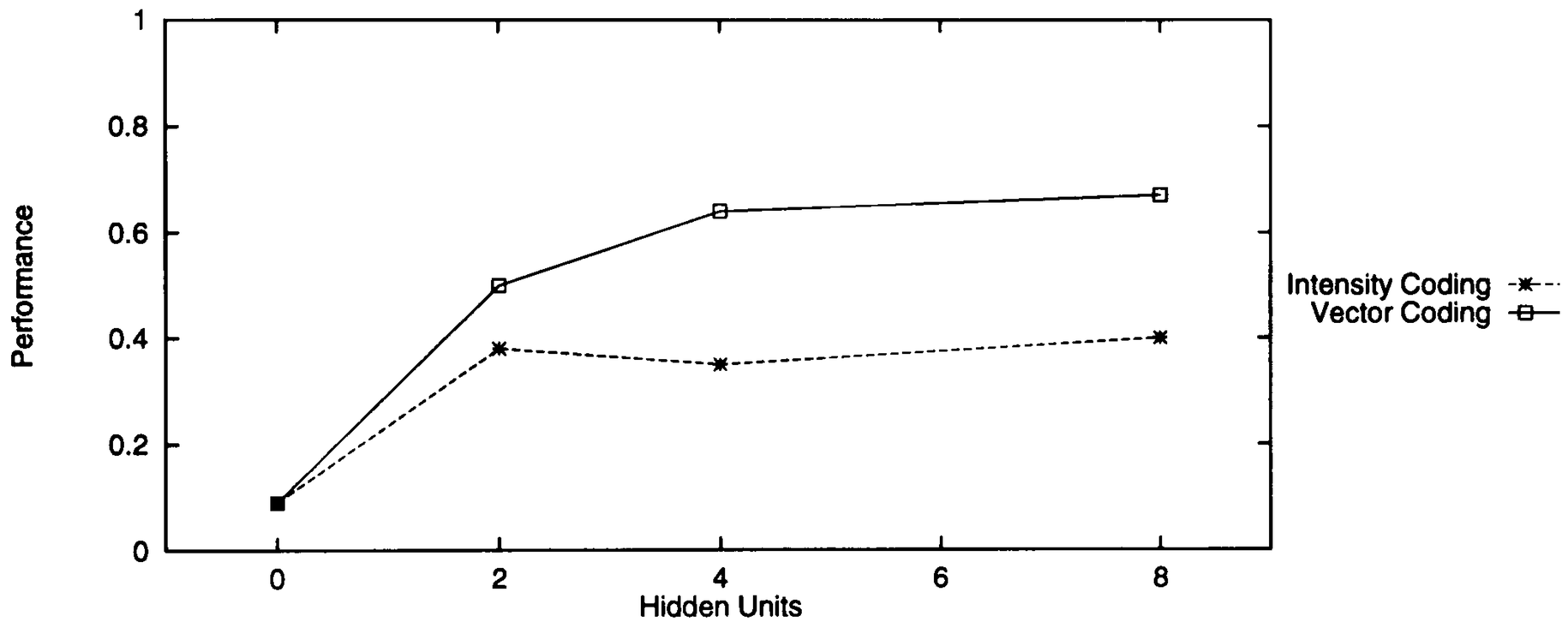


Figure 7.9: (a) The learning curves of 3 animats with 4 hidden units, learning task 1, with either intensity or vector coded sensory input. In each coding condition, curves for 3 animats are shown, with the mean performance over the previous 1000 trials plotted every 1000 learning trials.
(b) Mean performance after 50K learning trials, as a function of hidden unit size for task 1. In each condition, the mean performance of each animat over 1000 test trials without learning is averaged over 3 animats. All standard errors<0.06.

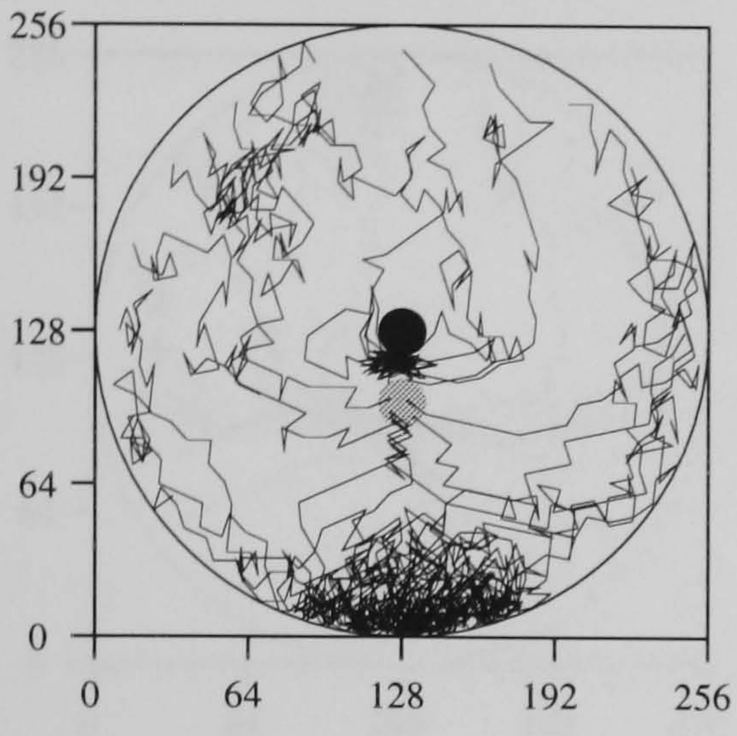
darker arrows of fig. 7.11b compared with fig. 7.10b).

When tested without a goal, this vector coding animat confines its search to the correct bearing and distance from the circle (fig. 7.11c), matching the gerbil search behaviour in this task. The same pattern of search in the goal location was found with all vector coding animats with four or more hidden units.

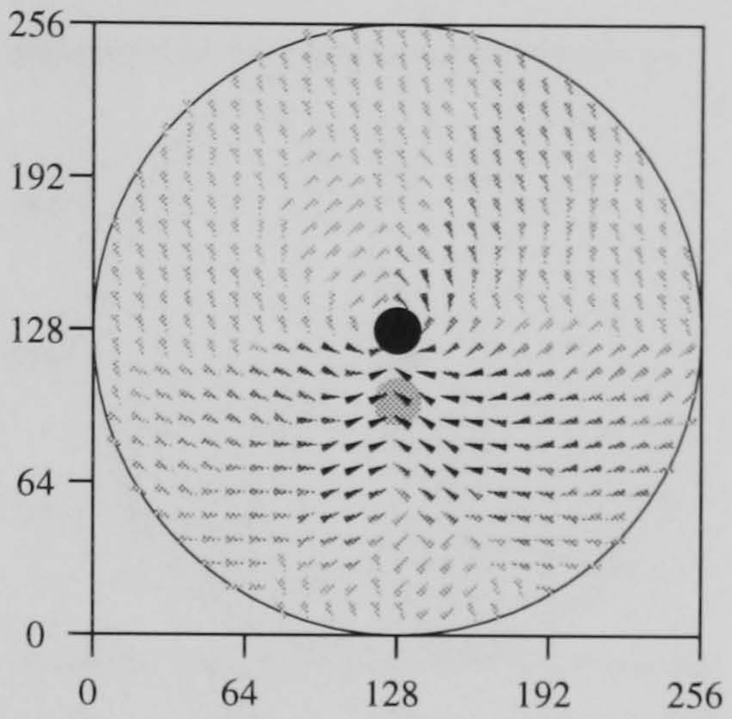
7.5 Task 2

The environment for task 2 consisted of two circles of radius 10, located at (-30,0) and (30,0), together with an invisible goal circle of radius 10 located at (0,-52) in coordinates relative to

(a) Example Paths.



(b) Behaviour.



(c) Search Histogram.

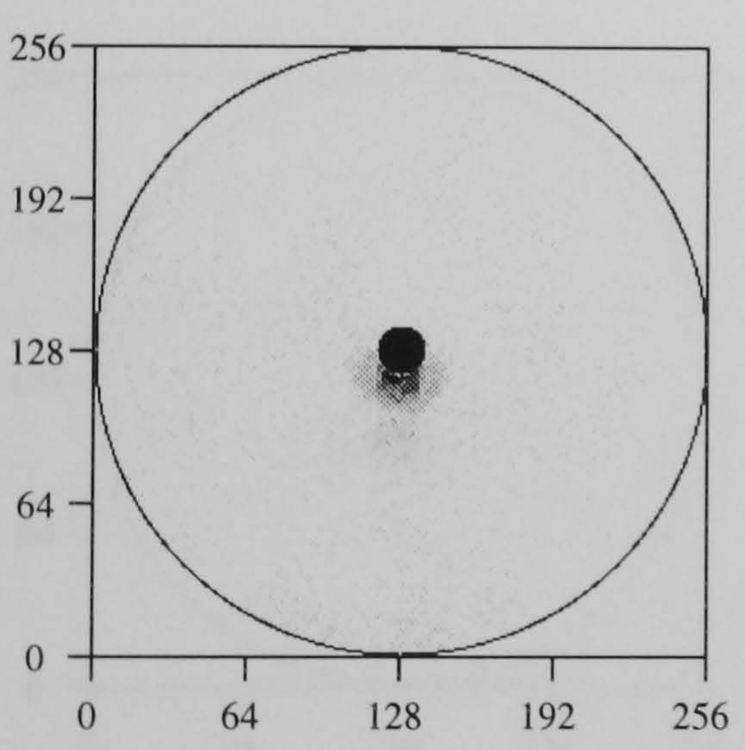
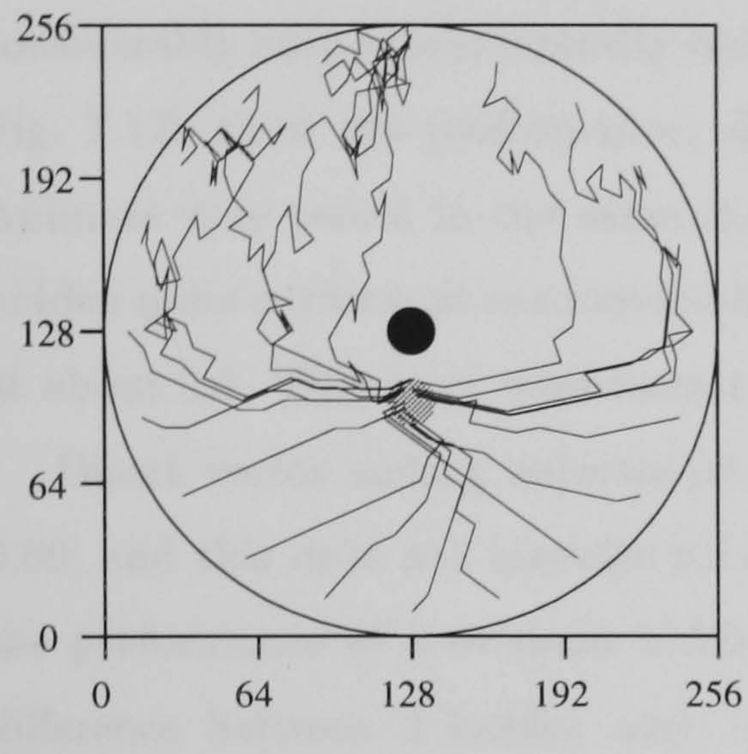
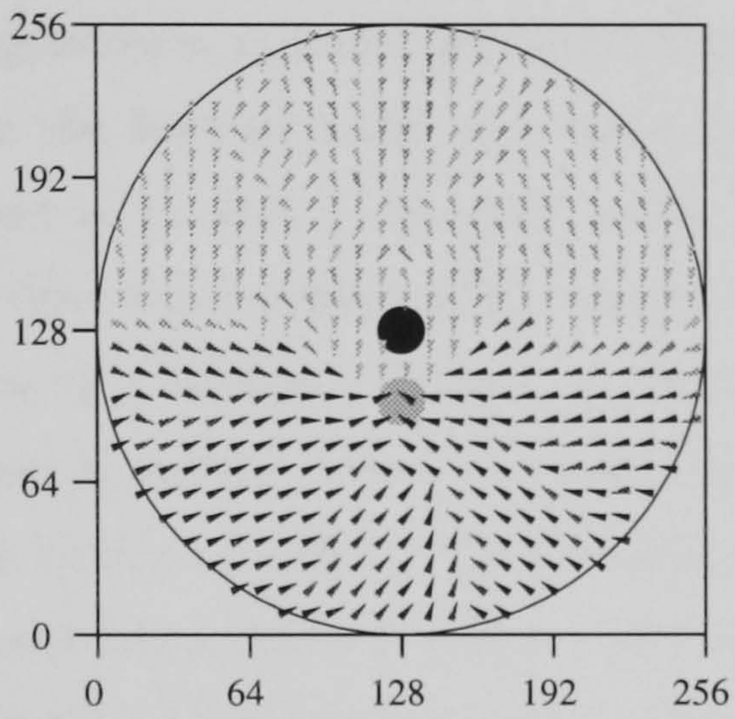


Figure 7.10: The behaviour and search pattern of a 4 hidden unit, intensity coding animat on task 1 after learning.
(a) 30 example paths of the animat to the goal, starting from locations around the arena edge. The invisible goal region is shown in gray.
(b) The animat's behaviour as a function of location. For each location in the arena, an arrow points in the direction in which the animat is most likely to move. The darker the arrow, the higher the probability that the animat will move in the arrow's direction when in that location.
(c) Search histogram. Animats were placed in the goal-less arena in a random location and moved for 50 time steps. This was repeated 100 times. The proportion of time spent in each location is plotted; the darker the pixel, the more time spent in that location.

(a) Example Paths.



(b) Behaviour.



(c) Search Histogram.

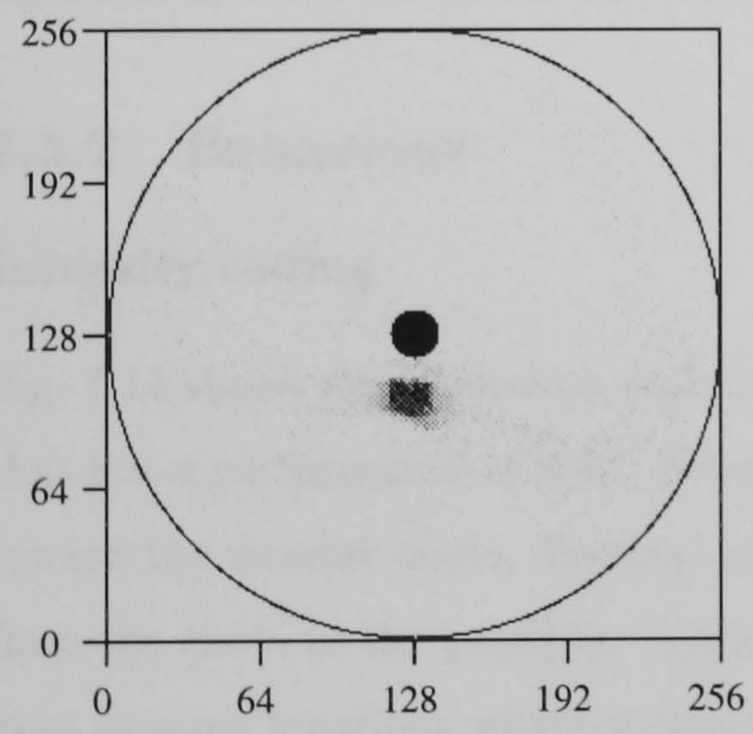


Figure 7.11: The behaviour and search pattern of a 4 hidden unit, vector coding animat on task 1 after learning.
(a) 30 example paths of the animat to the goal, starting from locations around the arena edge. The invisible goal region is shown in gray.
(b) The animat's behaviour as a function of location. For each location in the arena, an arrow points in the direction in which the animat is most likely to move. The darker the arrow, the higher the probability that the animat will move in the arrow's direction when in that location.
(c) Search histogram. Animats were placed in the goal-less arena in a random location and moved for 50 time steps. This was repeated 100 times. The proportion of time spent in each location is plotted; the darker the pixel, the more time spent in that location.

center of the arena. Animats were simulated with a fan in of 59 units, corresponding to 354 degrees.

7.5.1 Performance

Fig. 7.12a shows the learning curves of animats with 4 hidden units. Vector coding animats considerably outperform intensity coding animats, and converge to their higher performance faster. Fig. 7.12b show the performance, after 50k learning trials, as a function of filter network size. Animats were tested in the same manner as for task 1. Intensity coding animats with 2 or less hidden units perform at random levels. With 4 or 8 hidden units, intensity coding animats perform at about 0.5. This is an improvement on the best task 1 intensity coding performance of 0.4.

Direct vector coding animats perform at random levels, but with 2 hidden units perform at 0.69, and this does not increase with 4, or more, hidden units. 0.69 is around the same level as the performance of 4 or more hidden unit, vector coding animats on task 1. The performance difference between 2 hidden unit, intensity coding animats and 2 hidden unit, vector coding animats is significant (t-test: $t = 4.033$, $p < 0.02$). In task 2, not only do vector coding animats significantly outperform intensity coding animats, they do so with fewer hidden units.

7.5.2 Behaviour

Intensity coding

Fig. 7.13 shows the behaviour and search histograms for a 4 hidden unit, intensity coding animat that has a performance of 0.47. From most regions in the arena, this animat moves stochastically toward the nearest circle. Having arrived at the circle, the animat then moves in a straight path from the circle to the goal (fig. 7.13a). This two stage strategy for getting to the goal is exhibited even in some locations where a direct route to the goal would be quicker. For example, where the animat starts in the southwestern quadrant, it moves past the goal to get to the western landmark, only then to turn around and follow the straight path from the landmark to the goal. The tangle of paths around the right-hand landmark in fig. 7.13a suggests that the transition in behaviour from approaching the landmark to finding the path from landmark to goal is less well learned for the western landmark than for the eastern landmark. Animats move around the eastern landmark quite smoothly, until following the straight path from circle to goal. These behaviours can be seen more clearly in fig. 7.13b.

Figs. 7.13c)–d) show search histograms for this animat when tested with both the arrangement of landmarks experienced during learning, and modifications of this arrangement. In contrast to the situation in task 1, when tested with the learning arrangement, the animat searches at the goal location. Since all the intensity coding animats with 4, or more, hidden units search at the

goal location, it can be suggested that although a single landmark can only specify the bearing of a goal, two landmarks are sufficient for intensity coding animats to learn a location. The straight paths between the landmarks and the goal are also darkened in fig. 7.13c), reflecting the high proportion of time spent on these paths due to the behavioural strategy outlined above.

Testing this animat with a single circle (fig. 7.13c)) leads to unfocussed search close to, and more to the south of, the circle. This may have been expected from task 1 search behaviour, and implies that this animat uses both circles to guide the behaviour in the rest of fig. 7.13. When tested with the distance between the landmarks doubled, animats search at the location where the bearings of the landmarks are the same as during training (fig. 7.13d). Approximately the same search pattern was shown by all intensity coding animats with four or eight hidden units, and it is very different from that shown by gerbils (fig. 7.2). These results further support the suggestion that with intensity coding, animats learn only about the bearing of the landmarks and not about distance. They therefore search where the bearings match in fig. 7.13d), disregarding distance.

Vector coding

With vector coding, animats show a wider range of search behaviours than those with intensity coding. Fig. 7.14 shows the behaviour and search histograms of a two hidden unit, vector coding animat that has a performance of 0.70. This animat moves fairly directly toward the goal from anywhere in the arena except for the extreme northwestern region. The behaviour as a function of location plot (fig. 7.14b) supports this conclusion, with dark arrows covering most of the arena.

Fig. 7.14c shows that this animat searches at the goal location when tested with the two circles arranged as per learning, as do all vector coding animats with 2 or more hidden units. When tested with a single circle (fig. 7.14d), the animat searches mostly at the location with the same distance and bearing from the circle as was the goal from the western circle in the learning arrangement; suggesting that the animat learnt to move to the goal based solely upon this single landmark. When tested with the distance between the landmarks doubled (fig. 7.14e), the animat searches at the same location relative to the leftmost landmark. Though the leftmost landmark is the principal guide of the animat's search, the other landmark must have some influence, or else, in fig.7.14e, there would be as much search at the corresponding distance and direction from the landmark on the right.

Fig. 7.14 shows the behaviour and search histograms of a four hidden unit, vector coding animat that has a performance of 0.73. Like the gerbils of fig. 7.2, it searches mostly in two locations when faced with a single landmark (fig. 7.14d). From each of these locations, the landmark has the same distance and bearing as did one of the landmarks from the goal during learning. This animat also searches in an arc between these two locations at a constant distance from the landmark.

When the distance between the landmarks is double that during learning (fig. 7.15e), search

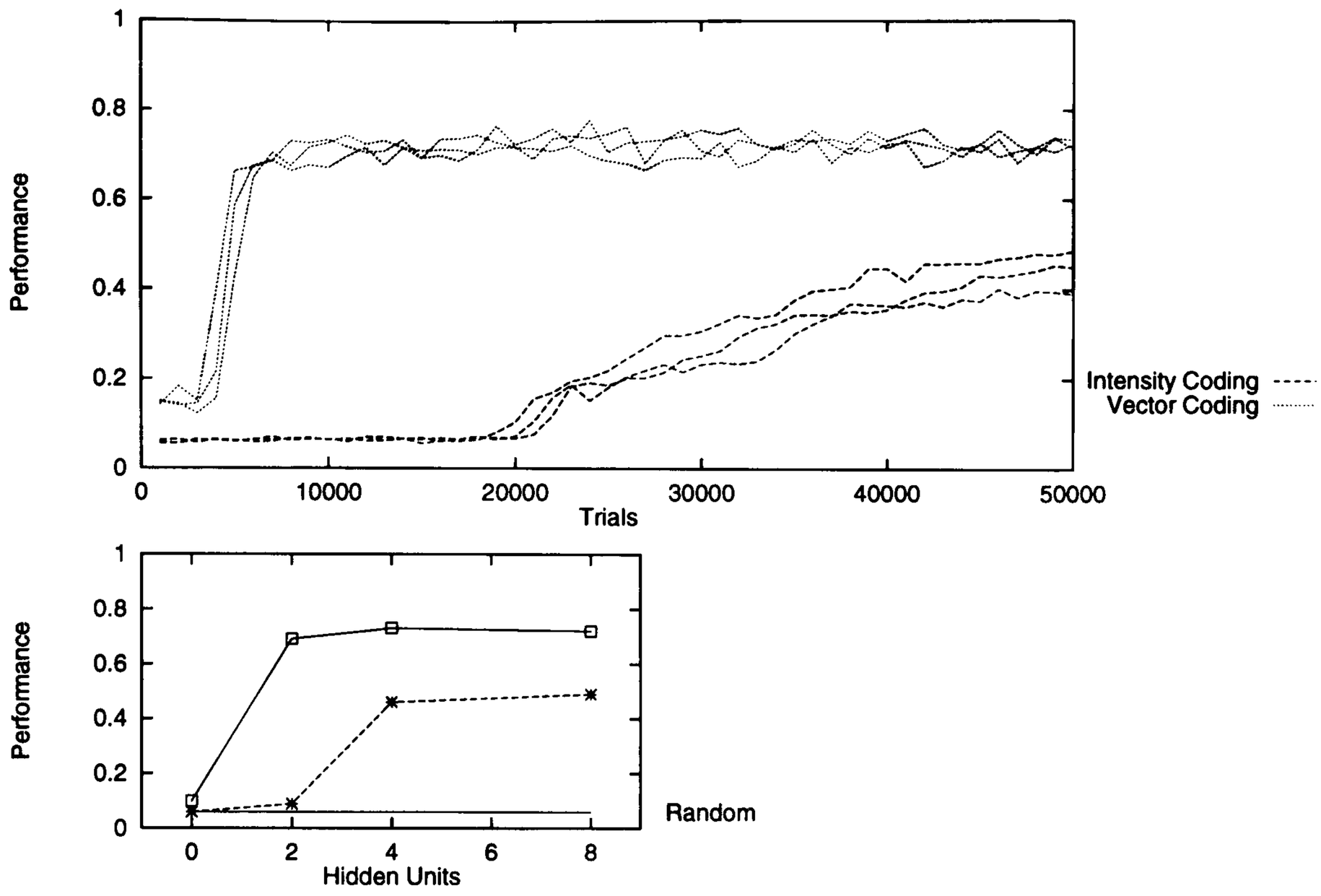
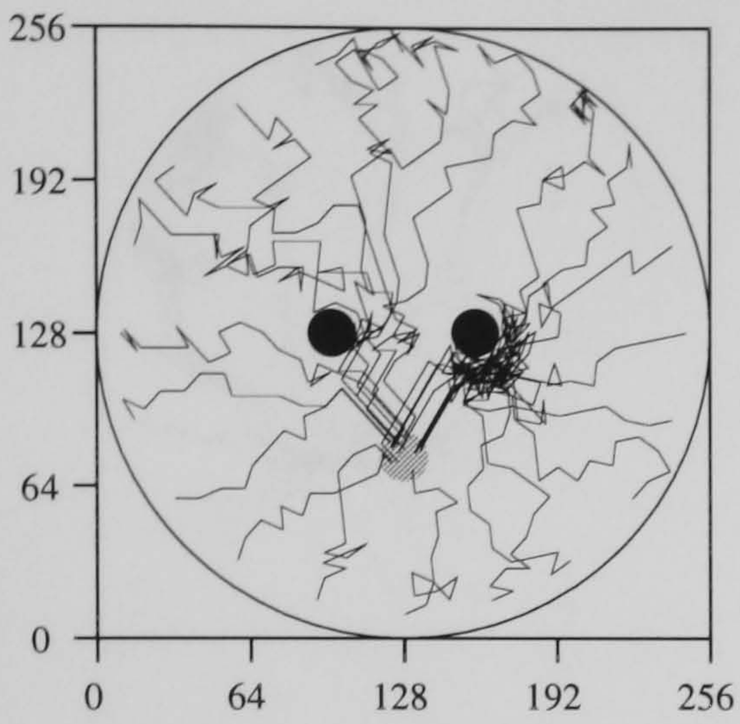


Figure 7.12: (a) The learning curves of animats with 4 hidden units, learning task 2, with either intensity or vector coded sensory input. In each coding condition, curves for 3 animats are shown, with the mean performance over the previous 1000 trials plotted every 1000 learning trials. (b) Mean performance after 40K or 50K learning trials as a function of hidden unit size for task 2. Each data point is the mean for 3 animats. Standard error for all data points is less than 0.03.

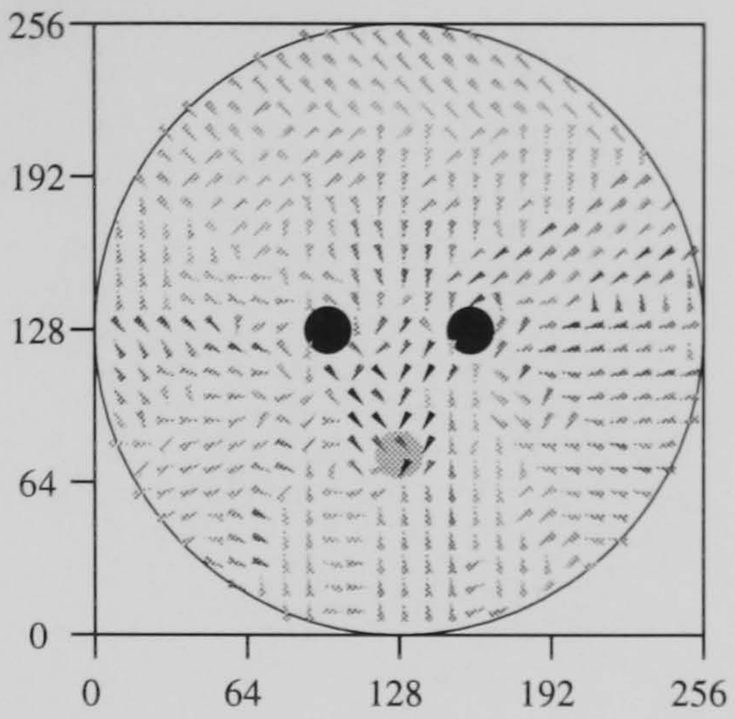
is mainly at a single location, which seems to be the one that best satisfies both the bearing and distance constraints from learning. In addition, some search time is spent at two outlier locations, both at the same distance and bearing as the goal from one of the landmarks during learning. This search pattern is different from that shown by the gerbils in Collett et al's (1986) study (fig. 7.2). Gerbils search at two discrete locations, rather than the single one of this animat. With one exception, all of the vector coded animats with 4 or 8 hidden units show roughly the same search behaviour. The exception is a 4 hidden unit animat that shows the same search behaviour as the two unit animat described above. The main source of variation amongst these animats is the relative amount of time spent searching at the single central location compared with the time spent at the 2 outlier locations, though in all, more time is spent searching within the landmarks.

Fig. 7.15f shows the search behaviour when the landmark array is rotated. Animats search in two locations each at the same distance and direction from the uppermost landmark as were the landmarks from the goal during learning. This arrangement was not tested with gerbils and so the search pattern of the animat is a prediction, and one that differs from the prediction of vector voting which predicts search in four locations.

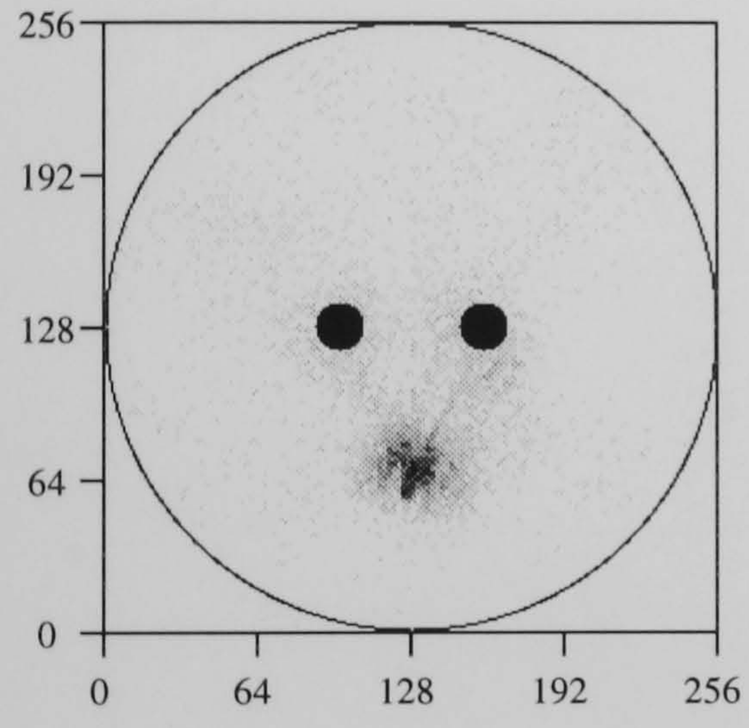
(a) Example Paths.



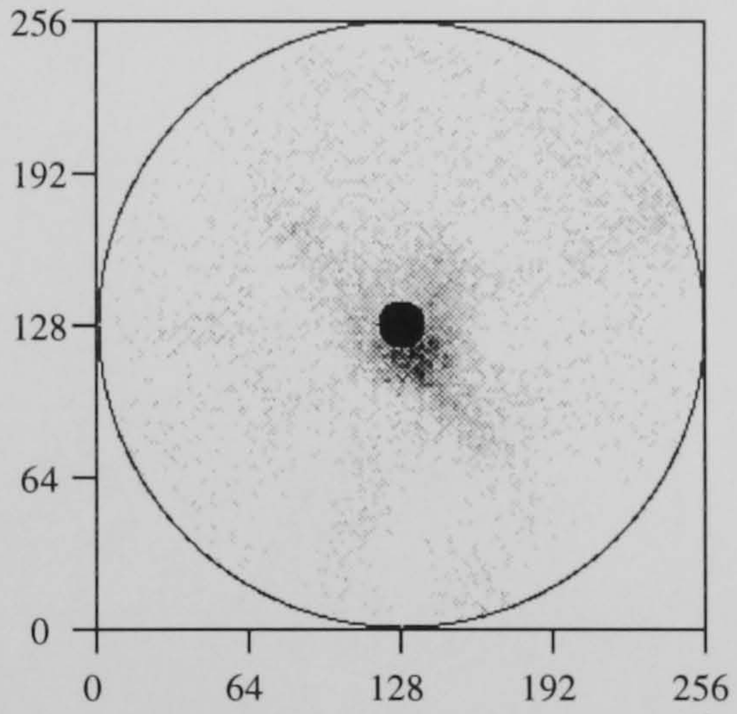
(b) Behaviour.



(c) Search Histograms.



(d)



(e)

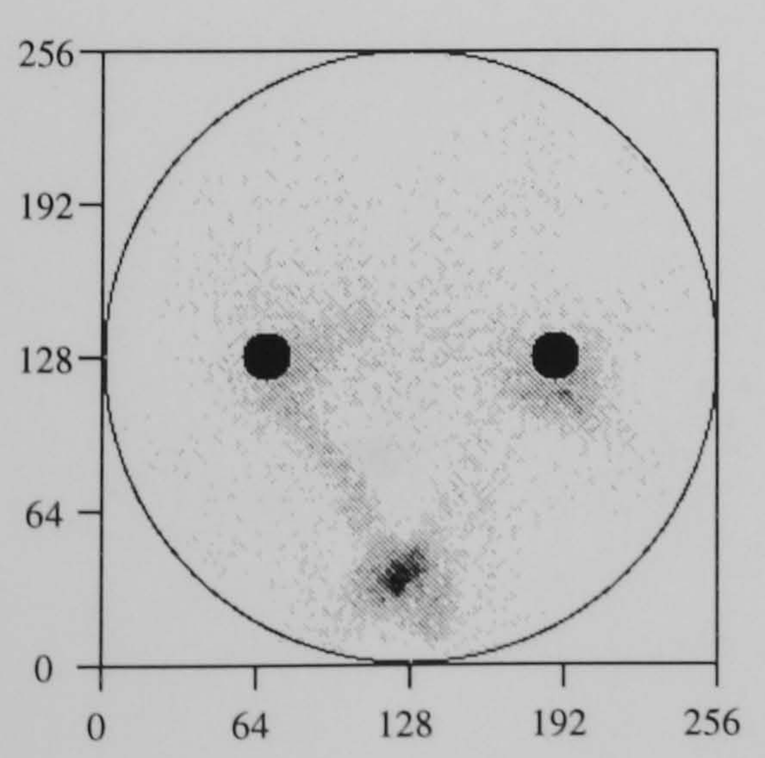
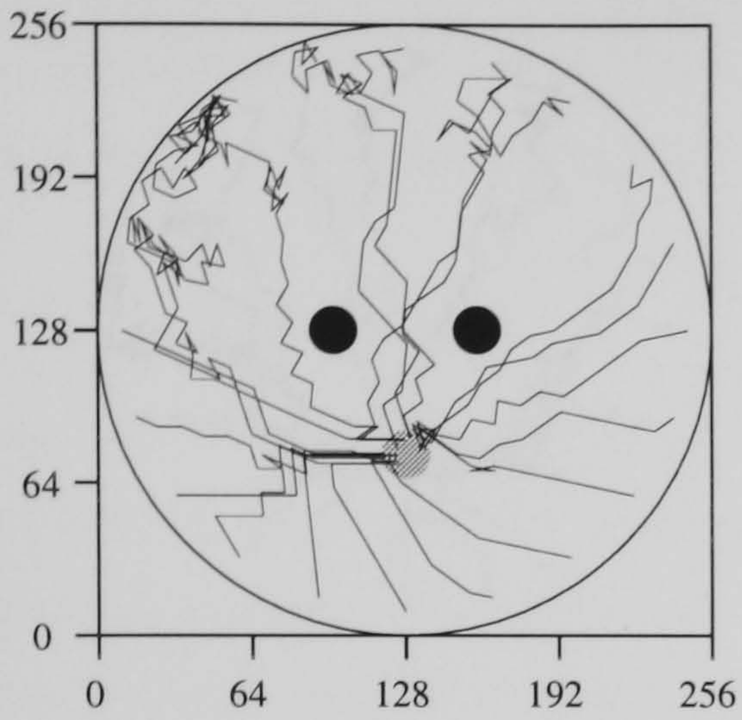
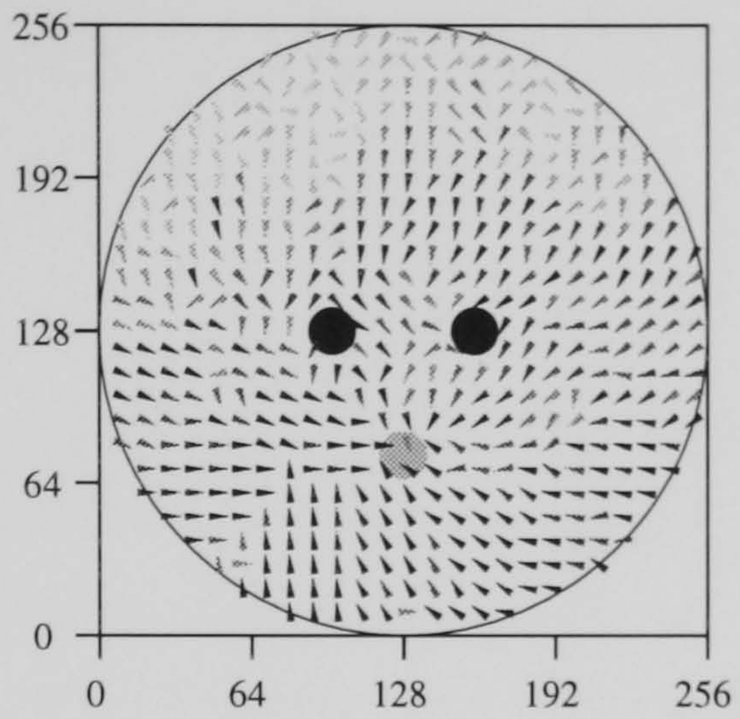


Figure 7.13: The behaviour and search patterns of a 4 hidden unit, intensity coding animat on task 2 after learning. Details as in previous figures.
(c) Shows the search histogram when tested with the environment the same as during learning.
(d) and (e) show search histograms in modified environments.

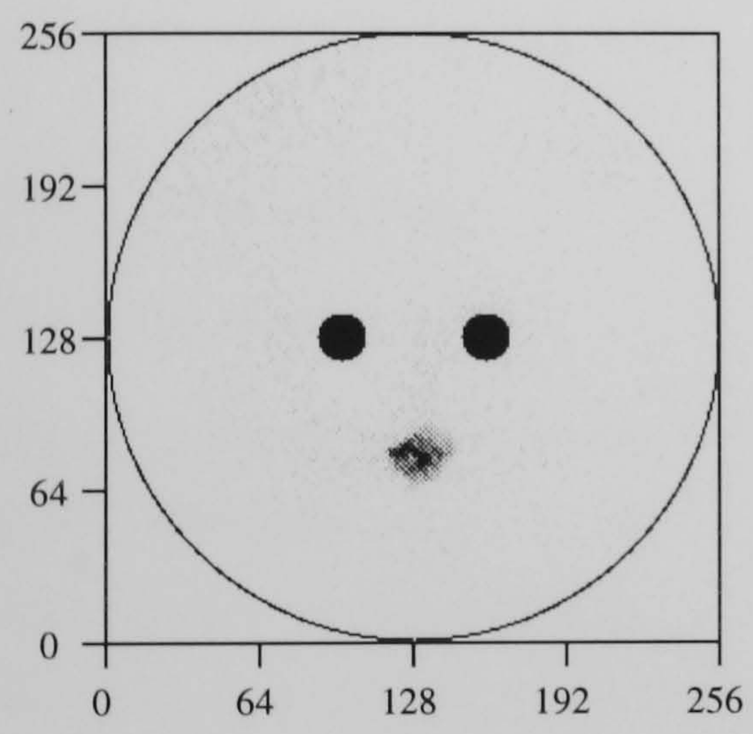
(a) Example Paths.



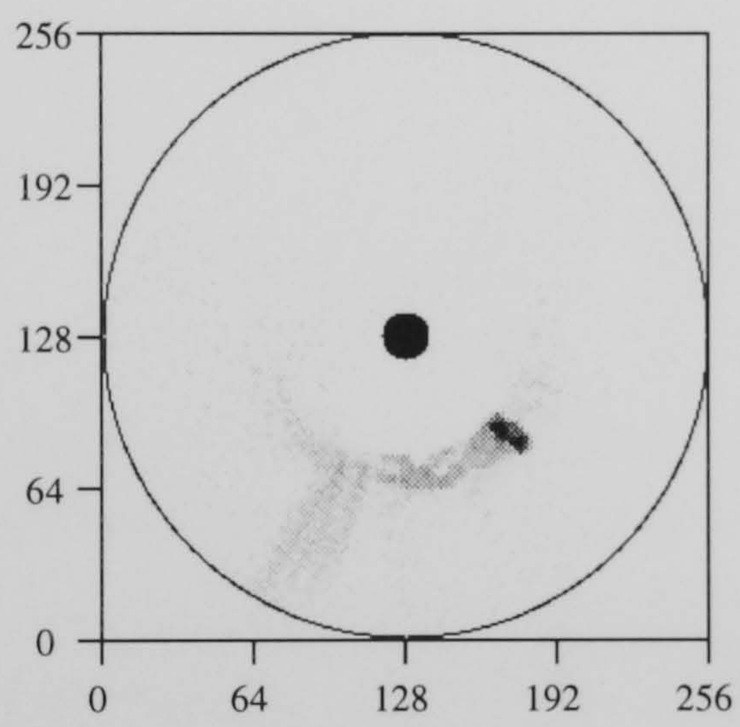
(b) Behaviour.



(c) Search Histograms.



(d)



(e)

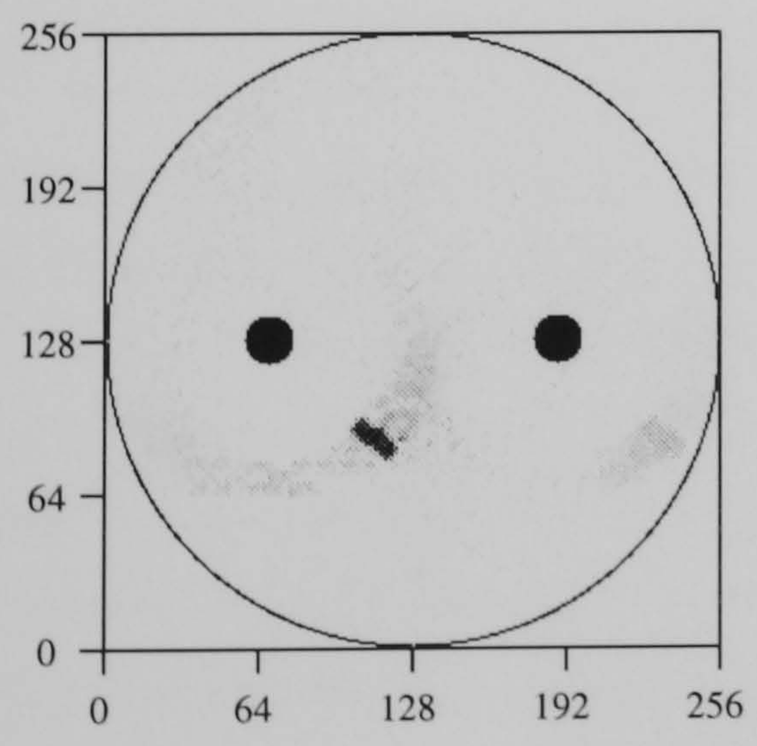
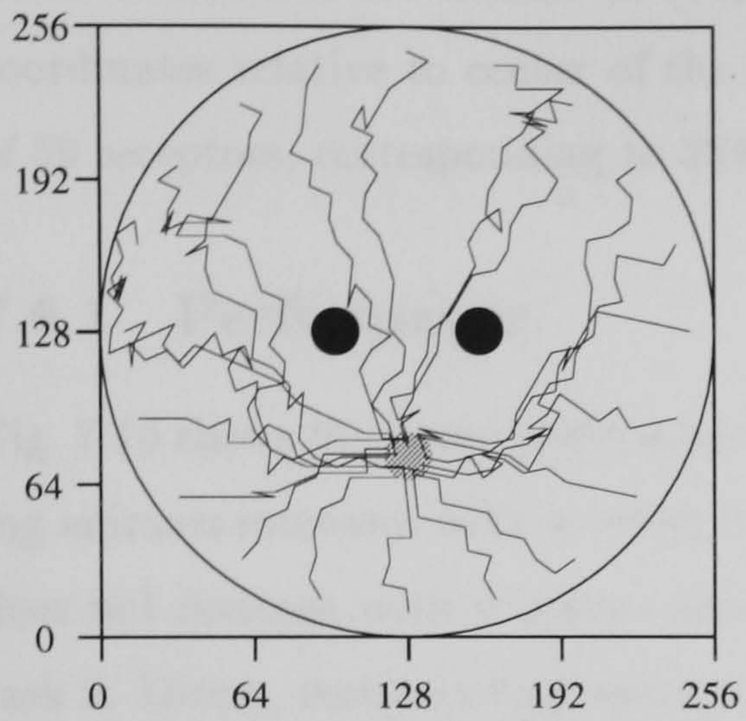
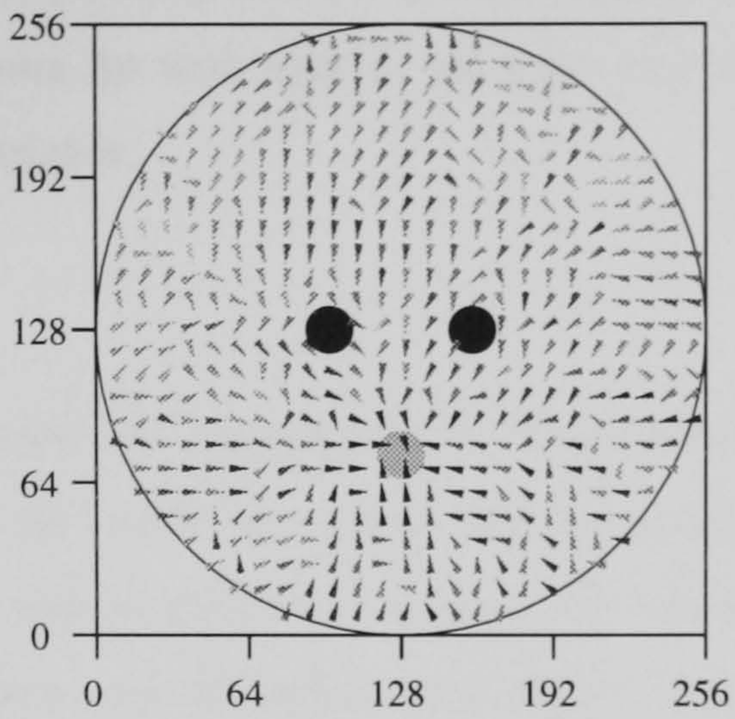


Figure 7.14: The behaviour and search patterns of a 2 hidden unit, vector coding animat on task 2 after learning. Details as in previous figures.
(c) Shows the search histogram when tested with the environment the same as during learning.
(d)–(e) show search histograms in modified environments.

(a) Example Paths.

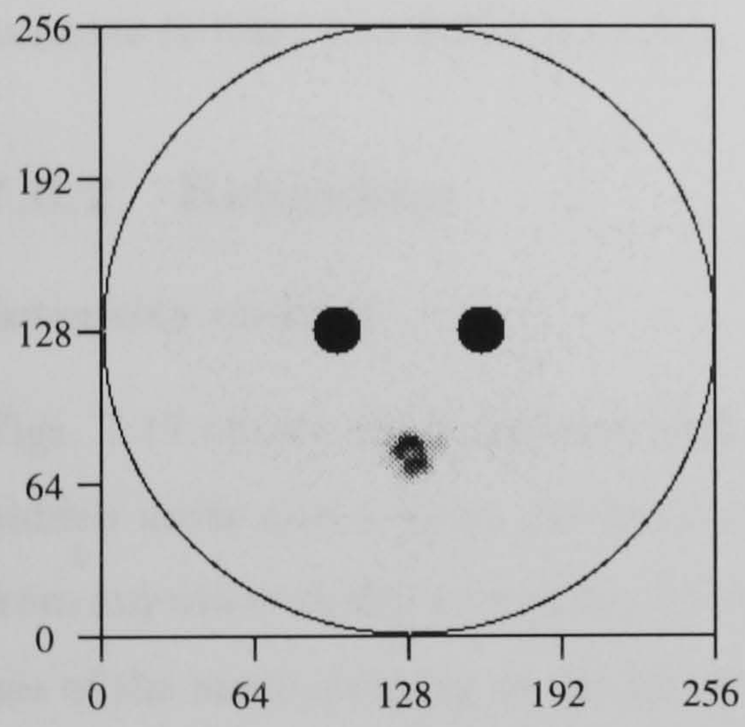


(b) Behaviour.

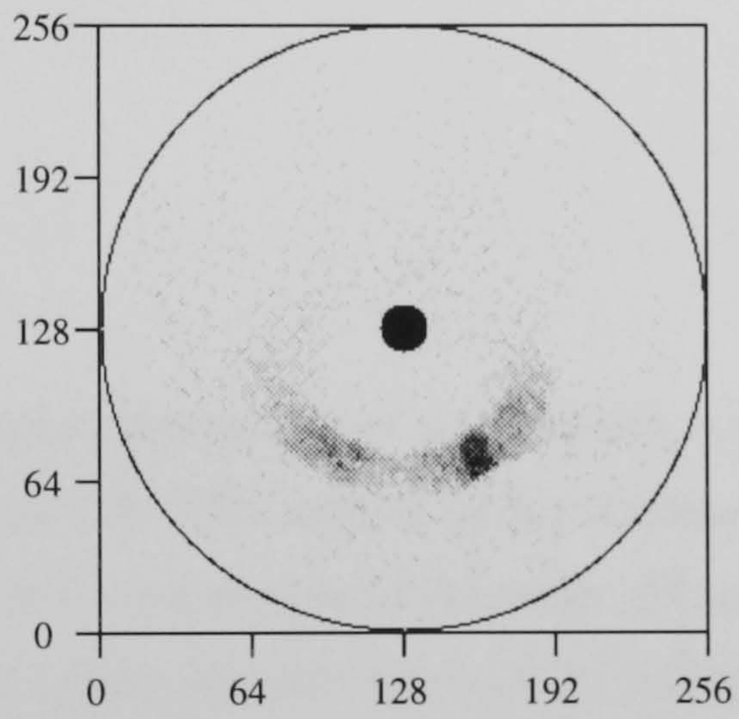


(c) Search Histograms.

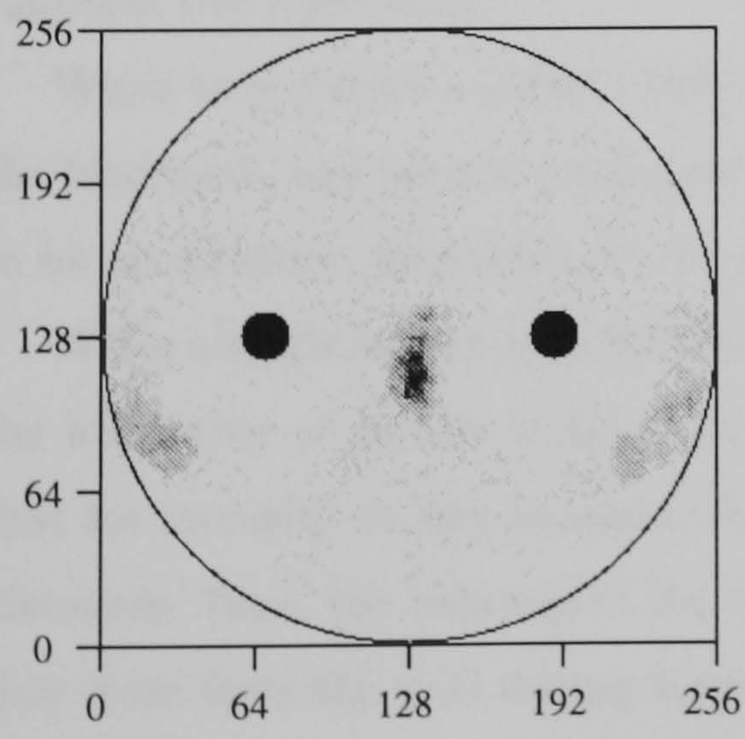
(d)



(e)



(f)



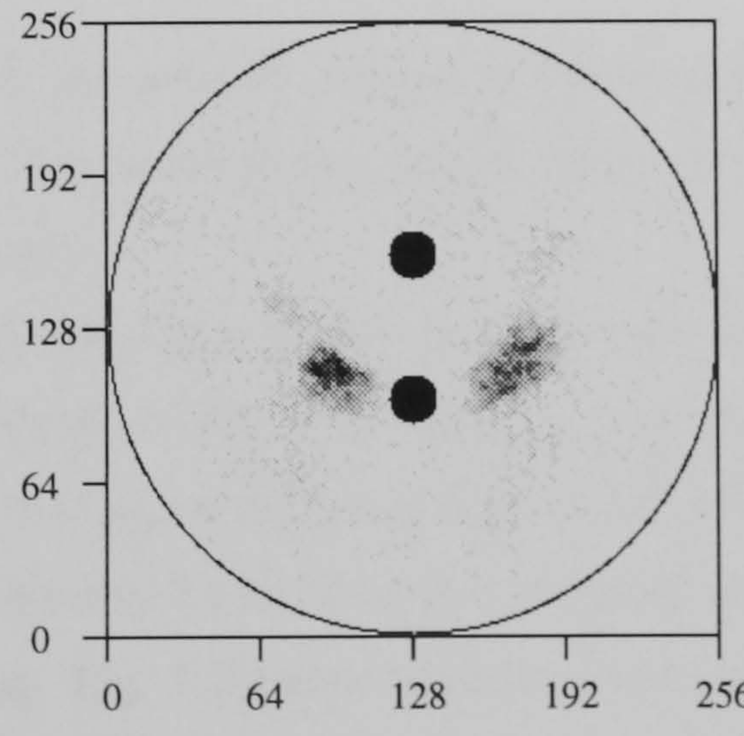


Figure 7.15: The behaviour and search patterns of a 4 hidden unit, vector coding animat on task 2 after learning. Details as in previous figures.
(c) Shows the search histogram when tested with the environment the same as during learning.
(d) and (e) show search histograms in modified environments.

7.6 Task 3

Three landmarks are located at (34,20), (-34,20) and (0,-40), and the goal is located at (0,0) in coordinates relative to center of the arena As with task 2, animats were simulated with a fan in of 59 receptors, corresponding to 354 degrees.

7.6.1 Performance

Fig. 7.16 shows the mean performance after 50k learning trials. The performance of intensity coding animats increases with network size up to 4 hidden units, which perform at 0.50. Performance does not increase with 8 hidden units, and is about the same as for intensity coding animats in task 2. Direct, vector coding animats have a mean performance of 0.78; animats with hidden units do not better this performance, and it somewhat higher than the vector coding performance in task 2. Direct intensity coding animats significantly outperform 4 hidden unit, intensity coding animats (t-test: $t = 8.251$, $p<0.01$).

7.6.2 Behaviour

Intensity coding

Figs. 7.17 shows the behaviour and search histograms of an intensity coding animat with four hidden units and a mean performance of 0.56. The animat moves stochastically toward the goal from anywhere in the arena (fig. 7.17a). For some regions of the arena, this animat appears to make use of the same strategy as the intensity coding animat examined for task 2 above; movement first to the nearest landmark, then following a straight path to the goal from the landmark. Fig. 7.17b supports this conclusion.

When tested with a single landmark, this animat engages in a very unfocussed search close to the landmark, and spends much time just wandering around the arena (fig. 7.17d). There appears to be no direction preference in the search.

When a single landmark is displaced, the search pattern is unaffected (fig. 7.17e). This matches the behaviour of gerbils in the same situation (fig. 7.3). It is an expected behaviour, assuming that for intensity coding animats, the bearing of the landmarks is far more important than their distances. Here, the bearings of the landmarks from where the animat is searching are the same as they were from the goal during learning. Fig. 7.17f shows search when the triangle of landmarks is rotated. In the center of the rotated triangle, the relative bearing of the landmarks matches that during learning, but their absolute bearings (with respect to the compass), have changed. Unlike gerbils, this animat does not search at the center of the rotated array, suggesting that the allocentric bearing of landmarks is more important to intensity coding animats than the relative

bearing between landmarks. At the three locations where this animat does focus it's search, two of the three landmarks in the rotated arrangement have about the same bearing as did two of the landmarks in the learning arrangement.

Fig. 7.18 shows the search histograms when additional landmarks are added to the learning arrangement in order to give the animat a choice between the triangle of landmarks from learning, and their inversion.. As with the gerbils of fig. 7.3, search is relatively unaffected by the additional landmark. This is to be expected given the previous result that intensity coding animats do not search in the center of the rotated triangle when presented in isolation.

Vector coding

Figs. 7.19 shows the behaviour and search histograms of a direct, vector coding animat that has a mean performance of 0.78. This animat moves fairly directly to the goal from any location (fig. 7.19a).

When tested with a single landmark, the animat searches mainly in 3 locations as shown in (fig. 7.19c). Each of these locations is at about the same bearing and distance from the landmark as was the goal from one of the landmarks in the learning arrangement. Approximately the same search pattern was shown by all vector coded animats. This pattern matches that shown by Collett et al's gerbils, as shown in fig. 7.3c.

When the distance of one of the landmarks from the rest is doubled (fig. 7.19d), the search location is not substantially affected. Thus, like the gerbils, this animat ignores an outlying landmark when two out of the three are in the usual place.

When the landmarks are rotated (fig. 7.19e), search is greatest at the center of the rotated array. In addition, this animat searches between the center, and three locations outside of the array, as did the gerbils in Collett et al's (1986) study. Each of these locations is at the same bearing and distance from 2 of the landmarks as was the goal during learning. The other direct, vector coding animats showed substantially the same behaviour, with the main variation being the extent of search in the 3 outlier locations when tested with the rotated arrangement. With hidden units, some of the animats search only at the center.

Fig. 7.18 shows the search histograms when an additional landmark is added to the learning arrangement. As with the gerbils of fig. 7.3, the search pattern of the animat is relatively unaffected by the additional landmark. As with the rotated landmark arrangement, the additional landmark introduces a location where the relative bearings between the landmarks, but not the allocentric bearings, match those from the goal during learning. Despite their behaviour when presented with only an rotated triangle of landmarks, animats do not search at the center of the rotated triangle when given a choice between this and a location where both the relative and absolute bearings and distances match those experienced during learning.

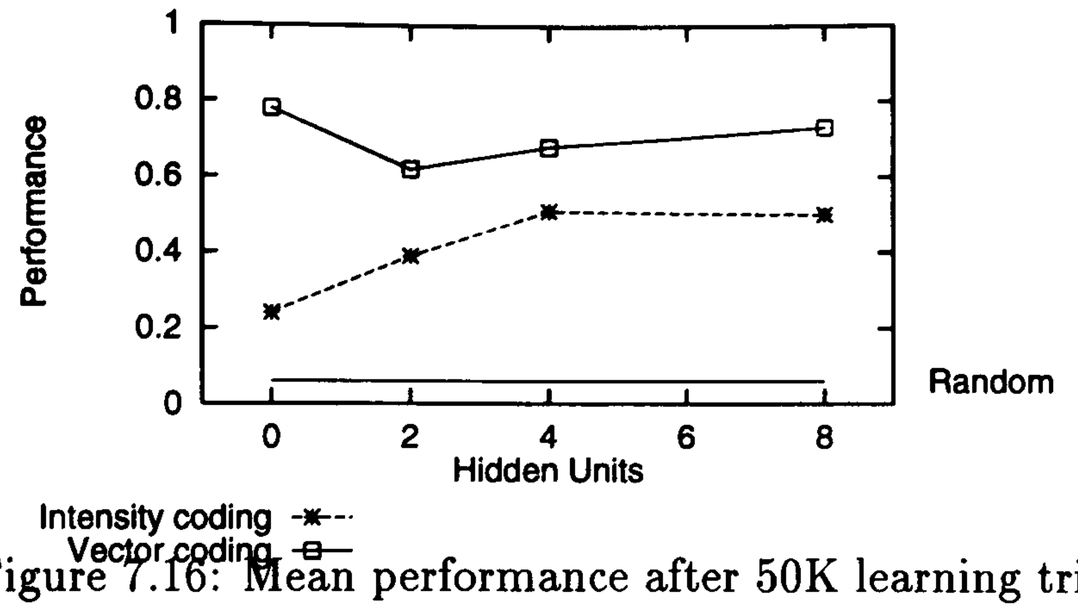


Figure 7.16: Mean performance after 50K learning trials as a function of hidden unit size for task 3. In each condition, the mean performance of each animat over 1000 test trials without learning is averaged over 3 animats. Standard error for all data points is less than 0.05.

The search behaviour of this direct, vector coding animat, matches that of the gerbils in Collett et al's (1986) study, for all the modified landmark arrangements (fig. 7.3). The same pattern of search, and hence close match to the animal behaviour was shown by all three direct, vector coding animats.

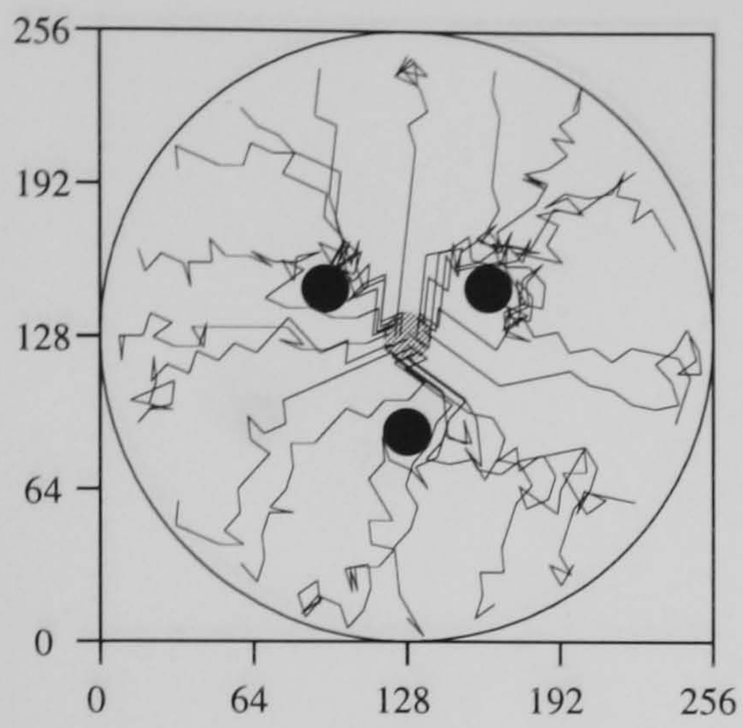
Behavioural predictions

Fig. 7.21 shows further search patterns for the direct vector coding animat found to replicate the gerbils search behaviour in task 3. With one landmark removed, the animat searches mostly where the goal was (Fig. 7.21a); this matches gerbil behaviour and is expected given the results with one landmark moved away as shown above. Figs. 7.21b–d are behavioural predictions from the animat, all of which differ from the predictions of vector voting. In fig. 7.21b, the landmark arrangement is both rotated and stretched; the animat searches at the center of this array. With the size of the triangle of landmarks doubled, animats search at the center (fig. 7.21c), with less search at three outlying locations. With the arrangement size doubled and rotated, animat search has an interesting pattern around the center of the of the landmarks and again search at three outlying locations (fig. 7.21d).

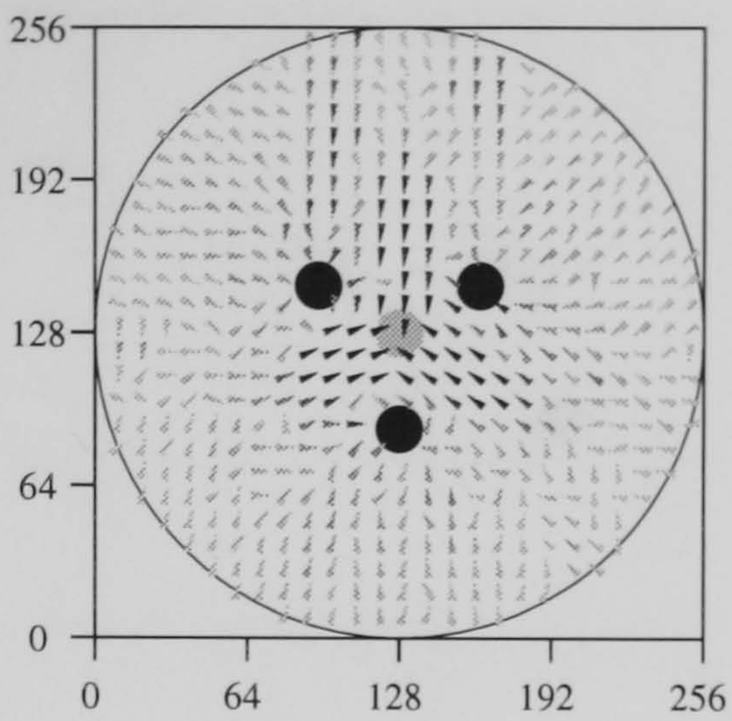
7.7 Discussion

This work shows that these simple and reactive convolution based animats, adapting by reinforcement learning, are capable of learning tasks analogous to those learnt by Collett et al's (1986) gerbils. Though the sensory array is modified from the visual array of previous chapters, animat's internal processing and learning parameters remained the same. Given enough hidden units, both intensity and vector coding leads to animats that perform at a level far above chance. These results are obtained with a relatively coarse coding of the sensory input: landmark bearing has a resolution of 6 degrees, the compass units of 24 degrees, and for vector coding, distance is split into one of just six bins.

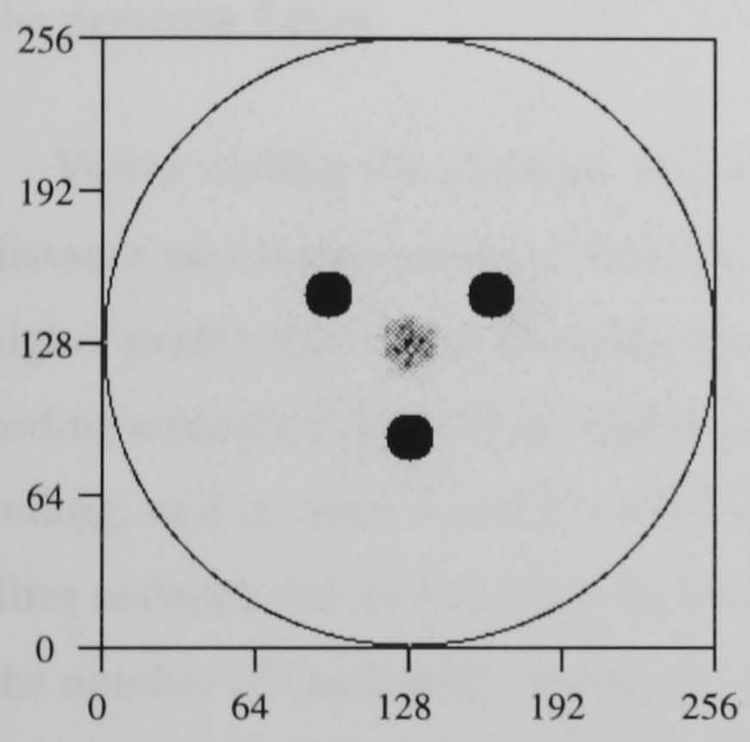
(a) Example Paths.



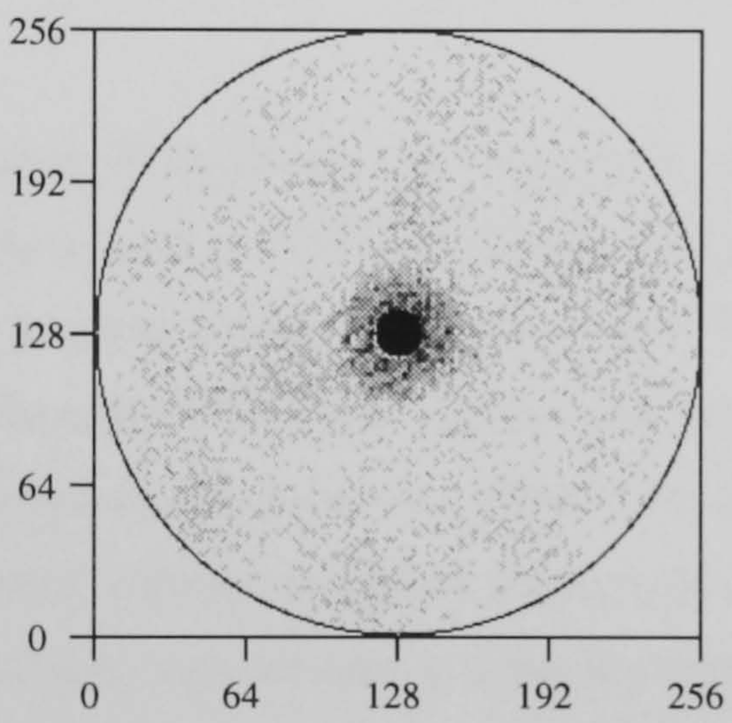
(b) Behaviour.



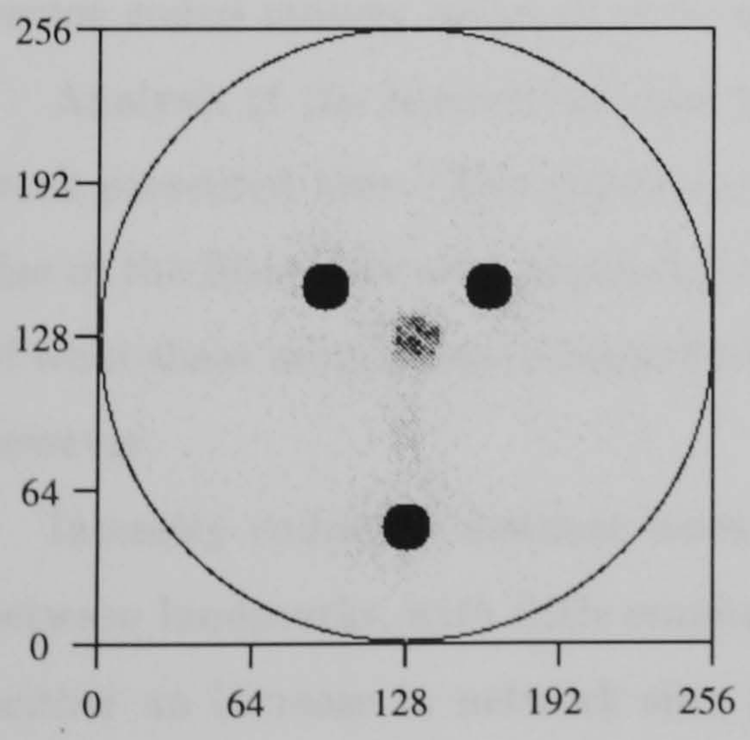
(c) Search Histograms.



(d)



(e)



(f)

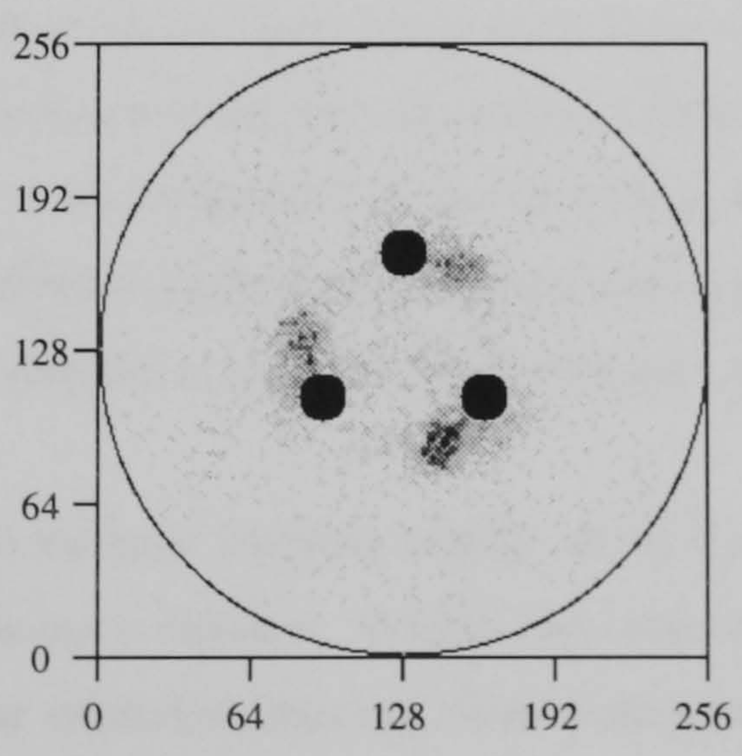


Figure 7.17: The behaviour and search patterns of a 4 hidden unit, intensity coding animat on task 3 after learning. Details as in previous figures.
(c) Shows the search histogram when tested with the environment the same as during learning.
(d)–(f) show search histograms in modified environments.

(g) Search histograms.

(h)

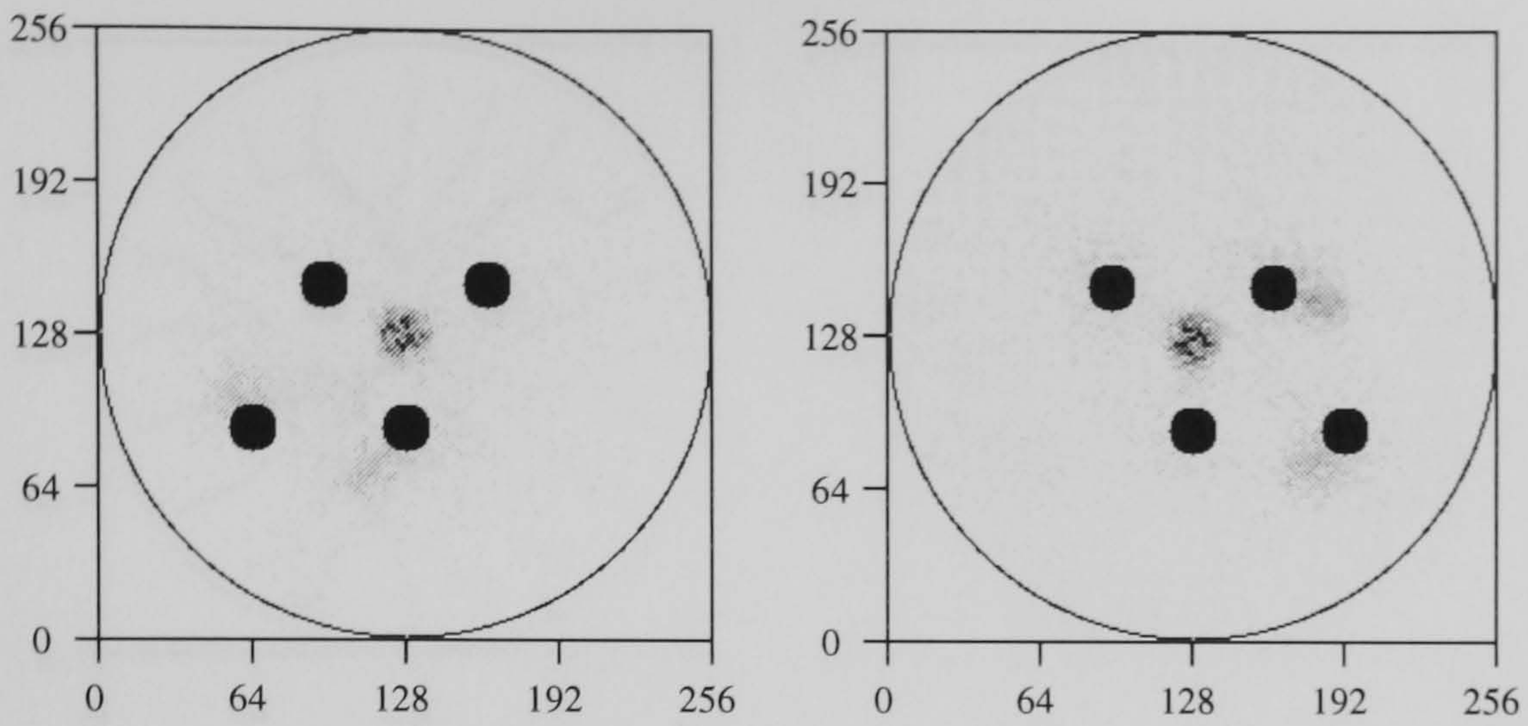


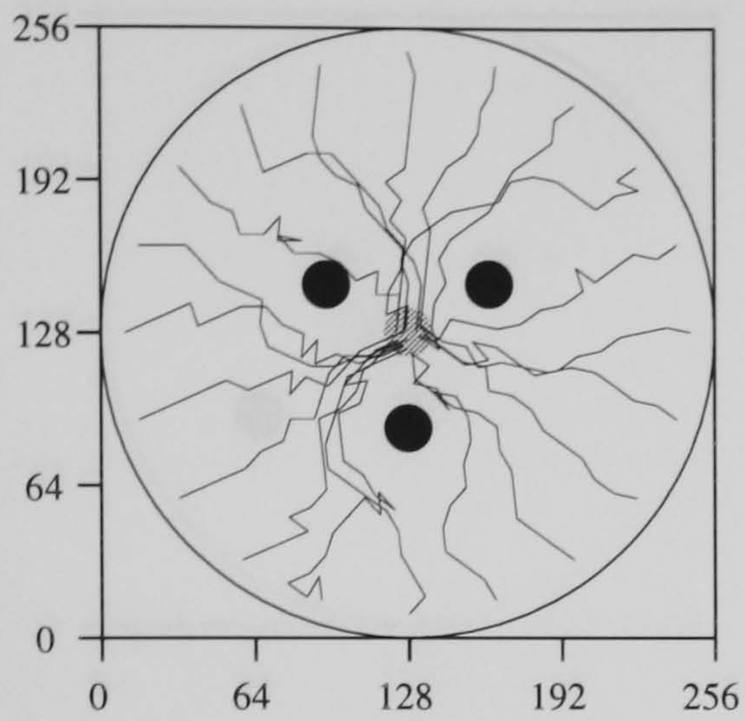
Figure 7.18: Further search histograms of the intensity coding animat, with 4 hidden units, from the previous figure.

Vector coding the distance and bearing of landmarks, in comparison with intensity coding of distance and vector coding of bearing, has a number of effects. Vector coding results in significantly higher performance after learning than intensity coding in all three tasks. In tasks 2 and 3, vector coding animats achieve their higher performance with fewer hidden units than required for intensity coding, and in tasks 1 and 2, vector coding animats have considerably shorter learning times. The filter network size required for vector coding animats to achieve maximal performance decreased as the number of landmarks increased: 4 hidden units for task 1, 2 hidden units for task 2, and direct filter networks for task 3. Intensity coding requires 4 hidden units to achieve maximal performance in all three tasks. This suggests that as the number of landmarks increases, the mapping from vector coded sensory array to movement array becomes computationally simpler.

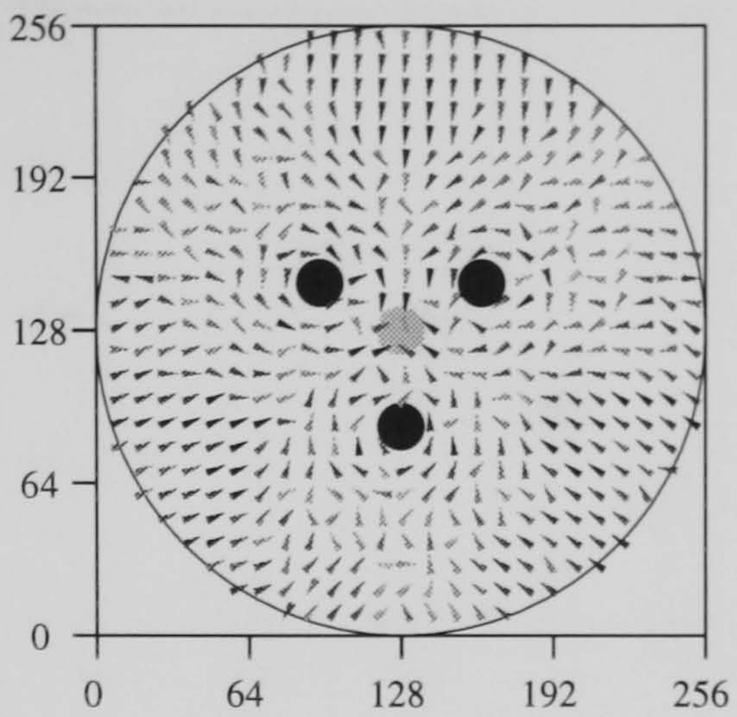
Analysis of the learned internal structure underlying the animats behaviour is lacking in the work presented here. The consistency of the behaviour of vector coding animats, and the small size of the filter networks required for efficient performance suggest that a detailed understanding of what these animats have learned to compute is possible. This remains a task for future research however.

Intensity coding of distance leads to animats learning mostly about the angular relationships between landmarks, with little emphasis upon distance. Within the range of values reported here, neither an increase in network size, nor extended learning trials makes much difference to this. When faced with single landmarks, intensity coding animats search directly around them. More evidence for their over emphasis on bearing information comes from task 2 with the distance between the landmarks doubled. In this situation, intensity coding animats search at the location in which the angle between the landmarks is the same as during learning. In task 3, intensity

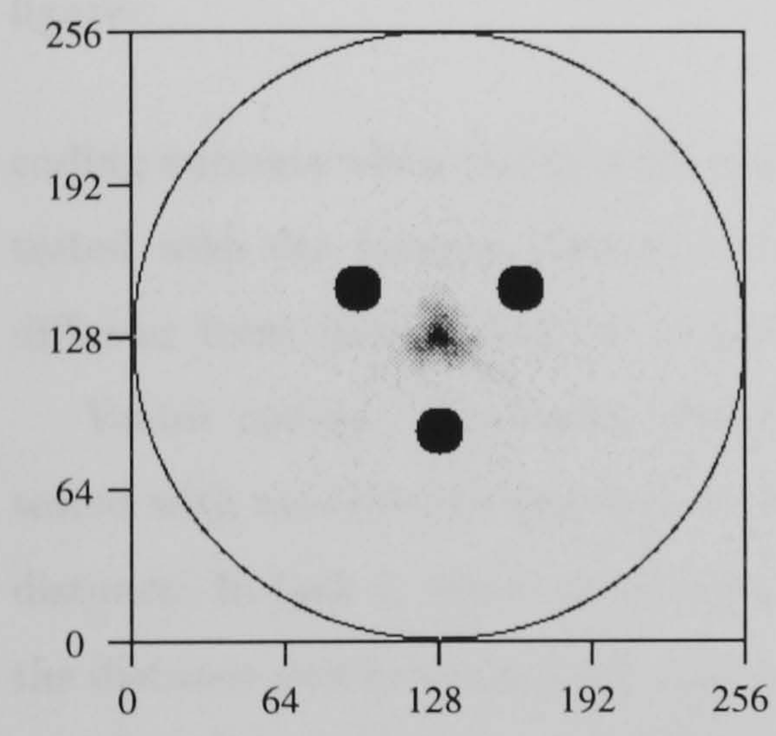
(a) Example Paths.



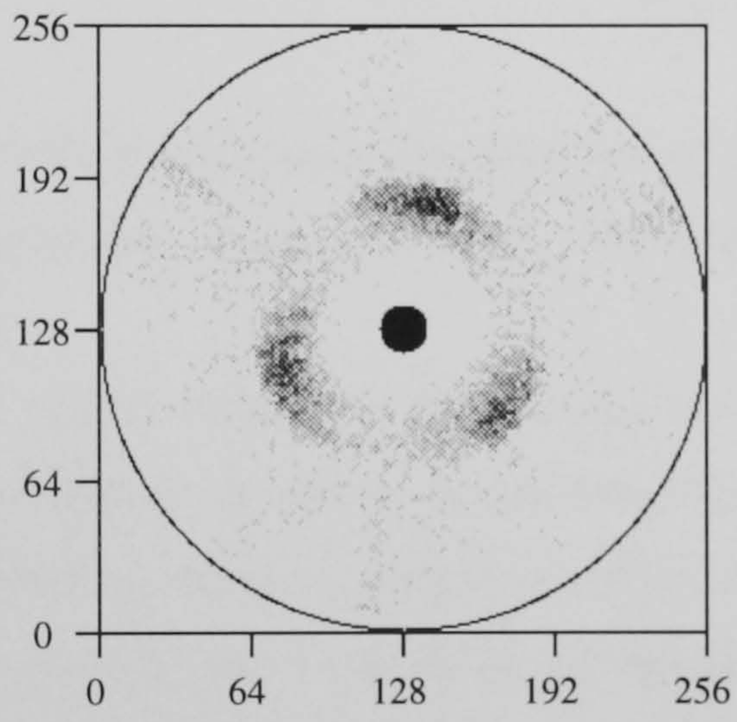
(b) Behaviour.



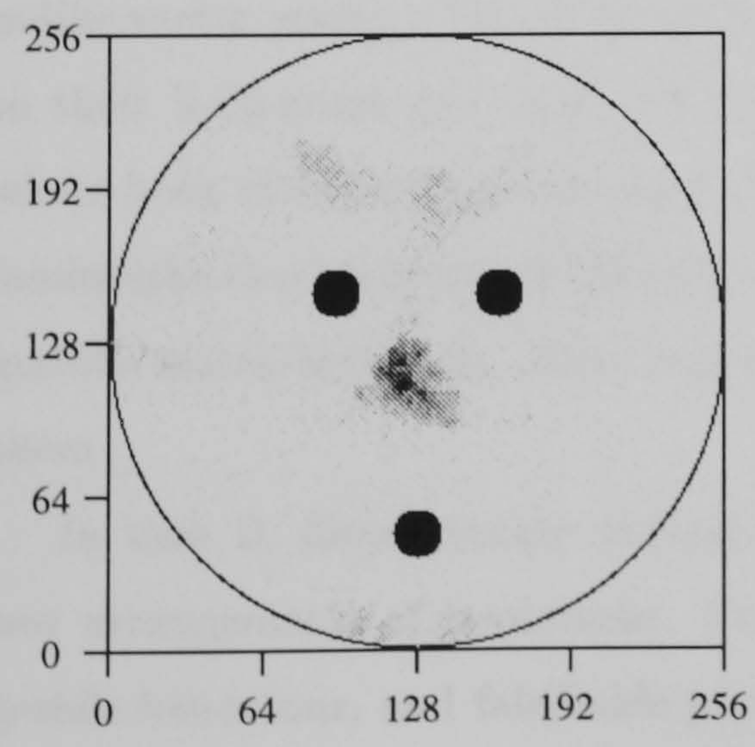
(c) Search Histograms.



(d)



(e)



(f)

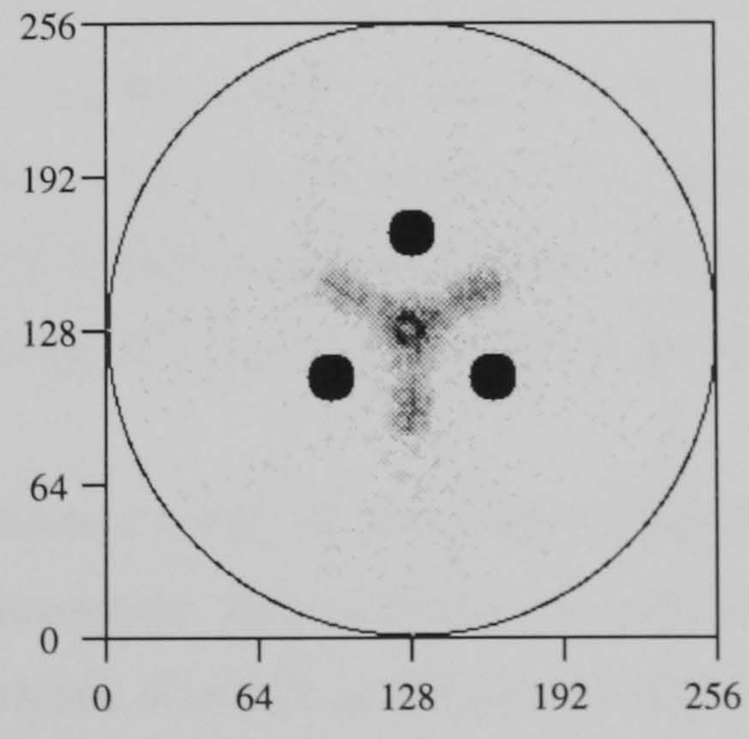


Figure 7.19: The behaviour and search patterns of a direct, vector coding animat on task 3 after learning. Details as in previous figures.
(c) Shows the search histogram when tested with the environment the same as during learning.
(d)–(f) show search histograms in modified environments.

e) Search histograms.

f)

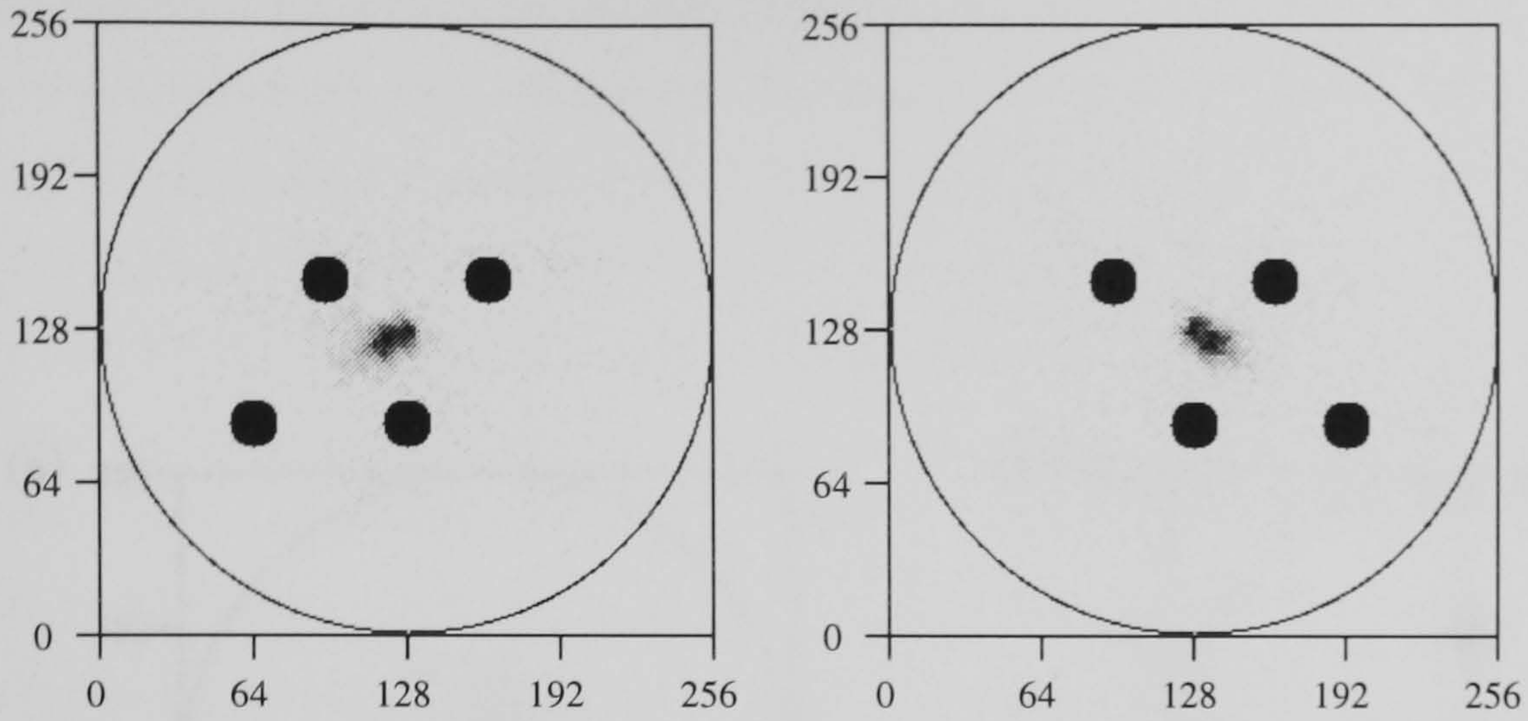


Figure 7.20: Further search histograms of the direct, vector coding animat from the previous figure.

coding animats when tested with a single landmark, again search close to the landmark, and when tested with the rotated triangle of landmarks, search near each. All these behaviours are very different from those exhibited by gerbils.

Vector coding, in contrast, leads to search behaviour close to the gerbil's when animats are tested with modified arrangements of landmarks. In task 1, animats search at the goal bearing and distance. In task 2, when tested with a single landmark, animats search in the two locations where the distance and bearing of the landmark match that of one of the landmarks during learning. This matches the gerbils behaviour and the prediction of vector voting. When tested with the distance between landmarks doubled, animats, searched mostly between the landmarks, like the gerbils and unlike vector voting. This implies they are sensitive to the relation between landmarks in addition to their individual locations. However, gerbils search at two discrete locations out of the four where both distance and bearing match; animats search mostly at the single location between the landmarks that most nearly satisfies both these constraints. Because of this failure to replicate the gerbil's search behaviour, these animats can be rejected as models of the computations underlying them.

In task 3, direct vector coding animats search in the same locations as the gerbils in all test arrangements of landmarks. Consequently, these animats can be posited as a model of the gerbils behaviour, and falsifiable predictions of search patterns in further modified arrangements have been produced to compare with those of gerbils. The close match to gerbil behaviour is surprising because of the coarseness of coding, and the simplicity and generality of the animats. The results seem to suggest that Collet et al's (1986) results do not necessarily reflect a specific spatial navigation system.

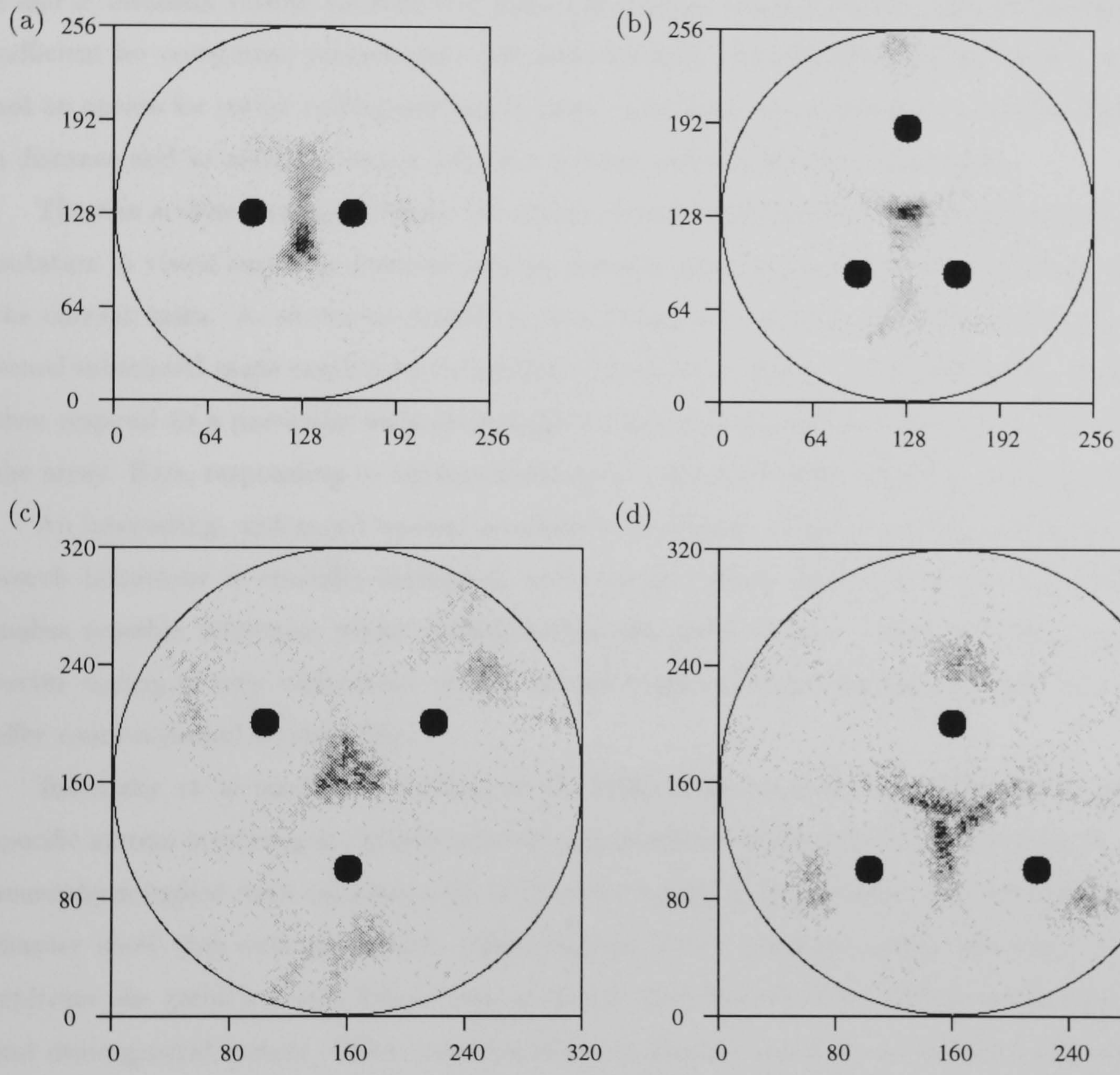


Figure 7.21: Further search patterns of the direct, vector coding animat of the previous figures. (a) matches the behaviour of gerbils, (b)–(d) are predictions.

Vector coding makes distance more explicit than does intensity coding, and this presumably underlies the superior performance of vector coding animats, and the smaller filter networks required to achieve peak performance. To respond to a landmark at a particular distance and relative bearing with vector coding is simply a matter of increasing a particular weight. With intensity coding, responding to the landmark at a particular bearing is as simple, but responding to a particular distance is a complex computation requiring, in general, two hidden units. Thus, in tasks 2 and 3, intensity coding animats rely upon the relative bearing between landmarks since this is sufficient for competent performance and easily learned with intensity coding. Such a strategy is not an option for vector coding animats because each input unit codes both a relative bearing and a distance and so animats cannot help but become sensitive to both dimensions.

There is a clear analogy between the utility of multiscale filtering for facilitating animat computation in visual tasks, as demonstrated in previous chapters, and the utility of vector coding in the current tasks. As shown in chapter 5, one of the properties of multiscale filtering is to make visual subtended angle explicit by converting it to a row within the multiscale array. Animats can then respond to a particular subtended angle by having large weights on a particular row within the array. Here, responding to landmark distance is similarly made easy by vector coding.

An interesting, and as yet unclear question is the extent to which animat replication of gerbil search behaviour is crucially dependent upon vector coding, and, if it is crucial, whether this makes possible inferences about coding within the gerbil brains. Gallistel (1990) argues that vector coding is very widespread within animal brains; perhaps simulations such as these may offer computational reasons why.

Touretzky et al model the Collett et al (1986) behaviours with a complicated and highly specific system involving an explicit internal representation of the spatial environment that models neurophysiological data on place cells within the rodent hippocampus. The simulations of this chapter show that such an explicit representation is not computationally necessary in order to replicate the gerbil's search behaviours in task 3, over the small set of test situations; a simple and more general system, closer to vision than navigation, can learn to behave in the same way. This finding raises the question of why do rodents have place cells if they are not computationally necessary? What computational and behavioural benefits do place cells give the gerbils? Insects can learn to move to a location specified by landmarks (eg Collett and Cartwright, 1983), and there is no evidence that they use the elaborate apparatus of place cells.

Unlike the gerbils, and Touretzky et al's model, the animats of this chapter, being reactive, are unable to complete trajectories if visual input is removed. In this, they behave similarly to insects.

Redish and Touretzky's (1997) model, because of it relation to vector voting, relies very heavily upon the allocentric bearing information provided by the head direction system. Regardless of

the number of landmarks, the only information from the visual array used to guide search is the allocentric vector to individual landmarks and the difference in bearing between them. In the animats here, egocentric visual information has priority, and is combined with allocentric compass information in a far more implicit way, in which animats learn to use whichever information enables them to successfully learn the task. Behavioural experiments in which animals learn goals defined by landmark arrangements which are rotated as well as translated between trials may allow separation of animal's use of visual and non-visual information in spatial navigation.

Chapter 8

General Discussion

Firstly, the main results of the thesis are reviewed, followed by consideration of some general issues raised by the research. Finally, some straightforward extensions of the research are outlined.

8.1 Review of the main findings

Repeated convolution by diverse filters partially characterises the early visual processing of animals as seemingly different as insects and mammals. The research in this thesis has shown that convolution by adaptive filters, modified by reinforcement learning, results in animats that can efficiently perform a range of simple visually guided tasks.

8.1.1 Approaching a solitary circle

In chapter 4, the goal was the region neighbouring a solitary circle in an otherwise empty arena, and contrast varied randomly between trials. When learning without sensory noise, direct animats performed highly regardless of coding. With either independent or coarse scale noise, rectified multiscale coding animats outperformed both intensity and multiscale coding animats. For both rectified multiscale and intensity coding animats, direct networks perform as well as those with hidden units in all noise conditions.

Consistently, intensity coding animats learned a balanced, step shaped filter which responds most strongly to the edges of the image of the circle.

The image of circle typically elicits activity at a range of spatial scales whereas noise causes activity at only a subset of scales. Rectified multiscale coding animats learn to exploit this to achieve higher performance by detecting the circle at the scales least affected by noise.

8.1.2 Learning visual subtended angle

In chapter 5, like chapter 4, the environment is empty except for a single circle. Here however, the goal region is where the angle subtended by the image of the circle falls within a particular range. Even without visual noise, rectified multiscale coding animats significantly outperform intensity and multiscale coding animats. Furthermore, rectified multiscale coding animats achieve peak performance with 2 hidden units compared with the 8 required for intensity coding. The difference between intensity and rectified multiscale coding animats increases in the presence of visual noise, regardless of its scale.

One result of multiscale convolution is that differences in subtended angle are converted into differences in which filter outputs are most active. Rectified multiscale coding animats learn to exploit this to achieve their higher performance by becoming sensitive to the relation between activity at different scales. Consistently, one hidden unit learns to respond to large subtended angle and the other to small subtended angle, and the output unit combines these to efficiently guide movement. Rectified multiscale filtering leads to higher performance because it makes subtended angle explicit, and so makes the mapping to useful output easier. Extracting subtended angle from the raw visual array is computationally much more difficult as is shown by the comparatively inferior performance of intensity coding animats. Animats having only a rectified single scale require 8 hidden units to approach the performance of those having multiscales, thus showing that multiscale filtering makes the relevant information more explicit than it is in any single scale.

Learning the same task in the presence of coarse scale visual noise drastically alters the computation learned by rectified multiscale coding animats. In this case, the same behavioural end, of moving to a goal subtended angle, are achieved by a different computational strategy involving the spatial pattern at fine scales.

8.1.3 Learning multiple subtended angles

Chapter 6 extends the results of chapter 5 by having a number of goal subtended angle ranges, one of which is chosen randomly on each trial. In addition to the heavier perceptual demands of this extension, it is a test of how well the learning algorithm is able to develop context dependent utility functions. The current goal is coded for animats by binary auxiliary units. Rectified multiscale coding animats greatly outperform intensity coding animats. With four hidden units performance approaches that of the 2 hidden unit best performing animats in the single goal task of chapter 5. Analysis of the animats after learning again demonstrates how multiscale convolution makes simpler the subsequent computations underlying efficient performance of the task.

8.1.4 Landmark learning

Chapter 7 simulates the gerbil landmark learning of Collett et al (1986). Gerbils, unlike insects, do not learn the angle subtended by a single landmark in this situation to guide movement, and so the visual array of the previous chapters is inappropriate to this case. Instead, a more abstract sensory code within the same framework is simulated, whereby each landmark causes activity in single direction in the 1-dimensional sensory surface, by an amount monotonic with both distance. Two ways of coding activity at a direction were compared; intensity coding, where a single continuous unit codes the value, and vector coding, where the value is coded by a number of binary units each of which is tuned to a non-overlapping range of values. Vector coding here, like the multiscale convolution coding of the previous chapters involves a considerable expansion of the sensory array.

Animats with vector coding learn to considerably outperform those with intensity coding on all three tasks simulated in this chapter. In the 2 and 3 landmark tasks, vector coding animats require fewer hidden units to achieve their best performance than intensity coding animats. With an equal number of hidden units, vector coding animats typically require considerably fewer learning trials to achieve asymptotic performance than intensity coding animats. Most importantly, the behaviour of vector coding animats, unlike that of intensity coding animats, closely matches the behaviour of gerbils when tested in a range of variations of the environment during learning. This is discussed more below.

8.1.5 Relation to animal research

As outlined in the introductory chapter, computational neuroethology allows a direct comparison between the behaviour of an animat and that of an animal. The closer the match between the environment and sensorimotor system of the animat and that of the animal it purports to model, the more weight can be placed upon any behavioural similarity.

There are typically many computational strategies for efficiently performing a task. As a consequence, if an animat learns to efficiently perform some task like an animal, it does not necessarily imply any similarity between the animats computational solution and that of the animal. If however, in addition to mimicking the animal at the learned task, its behaviour matches the animals when tested (without learning) in a variation of the learning environment, then the animat can more confidently be proposed as a model of the computations underlying the animals behaviour. The animat can be rejected as such a model when an environmental variation is found in which the behaviour of animal and animat diverge. Indeed, such environmental variations should be sought, since they define the limits of the animat as a behavioural model.

Animats modeling particular animal behaviours were simulated in chapters 5 and 7. In chapter 5, Cartwright and Collett's (1983) experiment showing that honeybees learn the angle sub-

tended by a landmark was simulated. Animats with both intensity and multiscale coding were shown to be able to learn to behave like the bees. This is not a very interesting result, because subtended angle is the only animat cue to distance, whereas for bees, it is one of a number of cues. Cartwright et al (1983), further show that after learning, bees move to the same distance when the landmark is replaced with only it's edges. When tested in this condition, all animats that had learned with either no sensory noise, or independent sensory noise, failed to generalise in this way, regardless of coding. However, rectified multiscale coding animats that learned in the presence of coarse scale sensory noise did generalise like the bees. Thus, all the other animats can be rejected as models of the computations underlying the honeybee's behaviour, even though they learned to perform the same behaviour.

In chapter 7, Collett et al's (1986) landmark learning experiments with gerbils were simulated. These experiments are useful because of the range of results in modified environments. Intensity coding animats, though they learned to behave like the gerbils when tested in the same environment as during learning, failed to match the gerbils behaviour when tested in modified environments. Intensity coding seems to lead to excessive focus upon landmark bearing rather than landmark distance. With vector coding of distance, animats behaviour closely matched that of the gerbils, and with three landmarks replicated their behaviour in all cases.

8.2 Multiscale filtering

Convolving the visual array with Laplacian of Gaussian filters at a range of spatial scales, followed by rectification greatly facilitates learning of the tasks in this thesis. Rectified multiscale coding animats learn to outperform intensity coding animats, and typically require less hidden units to achieve best performance. Multiscale filtering expands the visual array, and in doing so makes aspects of it explicit, thus making easier the subsequent computation. This empirically supports the arguments of Marr (1982), and Clark and Thornton (1994) amongst others that processing can be viewed as transforming the input array to one in which the transformation to output is as computationally as simple as possible.

8.2.1 Multiscale filtering in more general situations

The utility of rectified multiscale filtering of the visual array has been shown in the context of reinforcement learning and these particular tasks. This raises the question of whether such the same holds in more general cases.

A preliminary study compared intensity, multiscale and rectified multiscale coding in a supervised learning task using filter networks of the same structure as underlie the animats in this thesis. Instead of learning weights in response to the reinforcement signal, a pattern of activation

in the input array is presented together with a desired pattern of responses of the filter network output unit for each position. For the preliminary study, the filter network learned to detect a random target pattern embedded at a random position within a 1-D, randomly varying texture with the same statistics. Target and background patterns were obtained by convolving an array of uniformly distributed random numbers with a Gaussian. In support of the results above, rectified multiscale coding of the input array led to networks with far higher performance whilst requiring fewer hidden units. This preliminary study suggests that the area may be worth further research.

8.2.2 Why Laplacian of Gaussian filters?

In this thesis, Laplacian of Gaussian shaped filters have been assumed, based upon the animal research discussed in the introductory chapter. Although this shape is only an approximation of the shape of filters found within the early visual systems of animals, it is presumably close enough to make little difference given the simple images in this thesis. What has not been investigated is why this particular filter shape is so useful, and under what circumstances? To what extent do other filter shapes facilitate learning and in what contexts?

The computational arguments used to support particular shaped filters are based upon consideration of the computational properties of the filter (eg Marr, 1982; Watt 1988). For example, LoG filters are balanced, and respond strongly to edges etc. As shown in this thesis, computational neuroethology allows a different approach to this problem based upon assessing a particular set of filters according to how well animats, having their visual array convolved by such filters, can learn to perform a task. Thus, the utility of particular filters is first established behaviourally. A particularly interesting extension of this approach is to evolve the shape of filters with a genetic algorithm, selecting filters according to how well animats using them learn to perform. Over a number of generations, the set of filter shapes most suited to the task, in terms of facilitating learning, should be evolved. Having automatically generating the filters, and established their behavioural utility, the filters can then be analysed to determine their computational properties.

8.3 Learning time

One problem with this research is the number of trials required for learning. Animals typically require tens of trials rather than tens of thousands of trials, like the animats here, to learn similar tasks. More specifically, for the landmark learning tasks of chapter 7, Collett et al (1986) note that gerbils would typically require about 150 trials, spread over about a month, before a gerbil would run to the correct spot on release. Bees, faced with similar tasks, require around 30 to 50 trials before they reliably search in the right location (Cartwright and Collett, 1983). Interestingly, bees require fewer trials for learning 3 landmark tasks, than for learning with 1 or 2 landmarks; the

same pattern is shown by the vector coding animats in chapter 7: with one landmark, convergence is achieved within 20,000 leaning trials, and with 2 or 3 landmarks, convergence is achieved within 10,000. One relevant finding is that the expanded codings of rectified multiscale filtering in chapters 4–6, and vector coding in chapter 7 typically lead to shorter learning times than intensity coding.

As explained in chapter 3, the standard algorithms of Q learning and backprop were used. Many speedups and bells and whistles exist for these algorithms, especially for backprop, and they may be expected to significantly reduce the learning time. Speed of learning was not the focus of this research. At this stage in computational neuroethology, the most important thing is developing techniques for acquiring competent animats and examining their behaviour and computational structure. However, it must be said that there is no evidence that, even with the best of current algorithm speedups, animats learning time would decrease to anything like that of the animals. Furthermore, given the present learning algorithm, increasing the size of the arena (or decreasing animat step size), would be expected to lead to a disproportionate increase in time to converge.

Unlike the animats of this thesis, animals do not start the tasks with a random internal structure and behaviour. They can already see, and coordinate their movement and have personal experience in using vision for similar tasks. Additionally, learning to learn seems to be an important aspect of animal computation and Collett et al (1986) exploit this by retraining the same gerbils to gain an "enormous saving of time." (p. 836). Just as there is a selective advantage for animals that can move to their spatial goals more efficiently than their competitors, there will be a selective advantage for animals that can learn more efficiently than their fellows. The link between evolutionary adaptation of populations and individual learning has been explored by a number of researchers since Baldwin (1896) and Morgan (1896) initial hypotheses (eg Hinton and Nowlan, 1987; Parisi et al 1990; Acley and Littman, 1991; Floreano and Mondala, 1995). Hence, it seems reasonable to suppose that within the constraints imposed by the personal plasticity of particular animals, they will have evolved to efficiently exploit their potential for learning.

The animats in this thesis only learn a single task; an extension would be to compare the learning times of these, with those of animats that had previously learned similar tasks. It is not clear what would be the expected result.

8.4 Multi-purpose computations underlying learning?

Gallistel (1990, 1995) computationally analyses a range of learning behaviours including navigation experiments such as Collett et al (1986), examined in chapter 7 of this thesis. Based upon this computational analysis, Gallistel concludes that the specific computational demands of different tasks require task specific learning mechanisms. Thus, there is a specific computational learning

mechanism for learning navigation, a different one for learning temporal contingencies, and so on. Gallistel strongly argues that just as the kidney and liver are structurally different because they perform different functions, animals learning mechanisms are computationally specialised for solving particular kinds of problems. Gallistel sees no role for an underlying computational mechanism at a level higher than arithmetic and logic.

Throughout this thesis, the same underlying computation of convolution by adaptive filters is used. A range of tasks, form very simple visual tasks, such as approaching a single landmark, to complex navigation problems, such as moving to a location defined by the position of three landmarks, have all been shown to be learnable by this general convolution architecture with weights modified by reinforcement learning.

Before learning, animats are homogeneous. As a result of the different environments and goal regions, animats learn different weight structures within the general convolution framework. Analysis of the learned structure has shown that animats can be viewed as learning to implement computations reflecting the particular computational demands of their particular task. Undoubtedly, after learning, these computations are task specific, and can only be understood with reference to the task. However this was due to learning within a more general framework applied to a specific task, and not due to specialised, task specific, learning mechanisms. These results therefore show that, although reinforcement learning and convolution networks may not be all-purpose, they are multi-purpose and can learn to become highly specialised processing systems.

8.5 Further Work

8.5.1 Genetic algorithms

Genetic algorithms are a very general method of parameter modification based upon Darwin's principles of evolution by selection of heritable variation. These principles, and their application to neural network and animat research, were outlined in the introduction and literature review chapters. Genetic algorithms provide another way of adapting convolution network weights in tasks such as those of this thesis and would provide a very useful comparison to reinforcement learning. A population of animats with initially random filter network weights evolve under the pressure of an evaluation function defined simply in terms of how quickly animats move to the spatial goal.

GA's solve tasks directly in that animats are selected only for how well they do the task, regardless of how. Reinforcement algorithms are less direct in that they learn a mapping from sensory input to an estimate of how many steps from the goal that input implies the animat is. Given this function, movement is directed to lead to states that are nearer to the goal. Thus there

is an intervening computational step between the task and the behaviour that performs it.

This difference may lead to differences in computational strategy for solving the task. If both algorithms yield animats utilising a similar computational strategy, this would strongly suggest that the strategy reflects the computational demands of the task rather than the particular parameter modification algorithm. Differences indicate computations reflecting the particular type of algorithm and the breadth of possible computation.

8.5.2 Motor array coding

Within the convolution network animat design developed in this thesis, the output array, which stochastically determines movement, consists of a scalar for each direction in which the animat can move. Each number is the activation of the output unit of the filter network in that direction.

Given this array, the problem is to determine the direction in which to actually move. Here, the standard reinforcement learning solution is used: assigning probabilities to each direction according to how much the activation exceeds the mean activation. This method assigns probabilities only according to relative activations with no regard for the relative direction.

An alternative is to regard the motor array as a vector code and each activation as the length of a vector pointing in the corresponding direction. A resultant vector can then be obtained as the sum of these vectors, each weighted according to its length. This computation makes use of the vectorial form of the motor array, and its comparative utility can be behaviourally assessed by comparing the performance of animats using this computation with those using the non-vectorial computation of this thesis. In addition to the computational motivations of this operation, Gallistel (1990) and Georgopoulos (1995) argue that vector coding of motor output arrays, and computation of the resultant vector is a widespread feature of diverse animal computation.

8.5.3 Non-reactive agents through spatiotemporal filtering

A wide diversity of animals convolve their visual arrays with multiple filters having a scale and orientation specivity. In parallel, and concurrently with this, animals including vertebrates and invertebrates convolve their early visual arrays with filters having a temporal as well as a spatial structure. Such filters have been reported in animals as diverse as insects (eg. Horridge et al (1995)) and mammals (reviewed by Bruce and Green, 1985). These filters are sensitive to the spatiotemporal pattern of intensities within a local region of the visual array and are generally referred to as elementary motion detectors (EMD's). They respond most strongly to movement of visual contrast in a particular direction and at a particular speed (Horridge et al, 1995). The structure and computational properties of such filters have been investigated by Snippe (1991) and Horridge et al (1995) amongst others.

Sensitivity to movement is a component of a wide range behaviours, including time to contact (Lee 1980) and motion parallax for guiding movement (eg. Srinivasan et al, 1996) and judging distance (Srinivasan et al, 1989). As in the case with multiscale filtering, the evidence suggests commonality between the structure and computation of insect and mammalian spatiotemporal processing.

The utility of EMD's for animat behaviour has been shown by Francesinci et al (1992), who used the output of convolution of a visual array with EMD's to estimate distance and thus guide obstacle avoidance.

Simulating animats, and learning with EMD filters is a natural extension of the research presented in this thesis. Animats with a layer of EMD's, both in addition to, and in place of the multiscale LoGs could be simulated and compared with the reactive animats. There are a few alternative network architectures; these could be behaviourally assessed and analysed in the usual fashion. With EMDs, the non-reactive component of animat computation is provided by the visual filter rather than changes to the adaptive convolution network. Thus, the computational load is placed upon the filter rather than subsequent processing. Adding EMD type non-reactivity to the animats brings their computation nearer to that of animals, and allows simulation of a wide range of tasks having a non-reactive component.

A relevant first task for these non-reactive animats is to learn to move to a particular distance from a fixed radius circle. In chapter 5, reactive animats were shown to learn to use subtended angle to guide movement. With EMD and LoG filters, animats could either ignore the EMD array and just learn subtended angle (like the bees in Cartwright and Collett, 1983). Or, they could use the EMD array to learn a different strategy for estimating distance. In the latter case, the search distribution can be compared with that of the gerbils in the same task (Collett et al, 1986). In the happy case that they are similar, this opens the door to simulation of the multi-landmark gerbil experiments of Collett et al (1986) with a visual array rather than the distance sensitive sensory array of chapter 7.

8.5.4 2-dimensional visual arrays

The most interesting potential extension of this research is to 2-D visual arrays. This thesis has demonstrated the computational utility of expanding a 1-D visual array to a 2-D array by convolution with filters at multiple spatial scales. Animals have 2-D visual arrays, and, as outlined earlier, a wide range of evidence suggests that the early stages of both invertebrate and vertebrate vision involves convolution with filters having both a specific scale and orientation. In this case, the 2-D visual array is expanded to a 4-D array because each filter is specified by both an orientation and a scale. Rectification yields two, 4-D arrays. Fig. 8.1 shows an example where an image of a flower is convolved with difference of Gaussian filters at 3 spatial scales (columns) and orientations

of 0, 45, 90 and 135 degrees (rows). Flowers are a particularly interesting example since their shape has presumably evolved to reflect the visual system of insects (and vice-versa) because of the evolutionary pressure for them to be as visible as possible to the insects that frequent them.

The generalisation from the research of this thesis to the 2-D case is straightforward: the 4-D multiscale and orientation array is convolved with a filter network to yield a 2-D motor array to stochastically determine the direction of movement. The computational neuroethology approach permits the behavioural assessment of the utility of such a scheme by comparing the performance and behaviour of animats learning (or evolving) with and without the filtered visual array. Furthermore, a wide range of experiments in which animals learn to discriminate between patterns and are then tested on their preferences with variations of these patterns would then become open for modeling (eg. Zhang et al, 1992; Srinivasan et al, 1993; Srinvivasan, 1994).

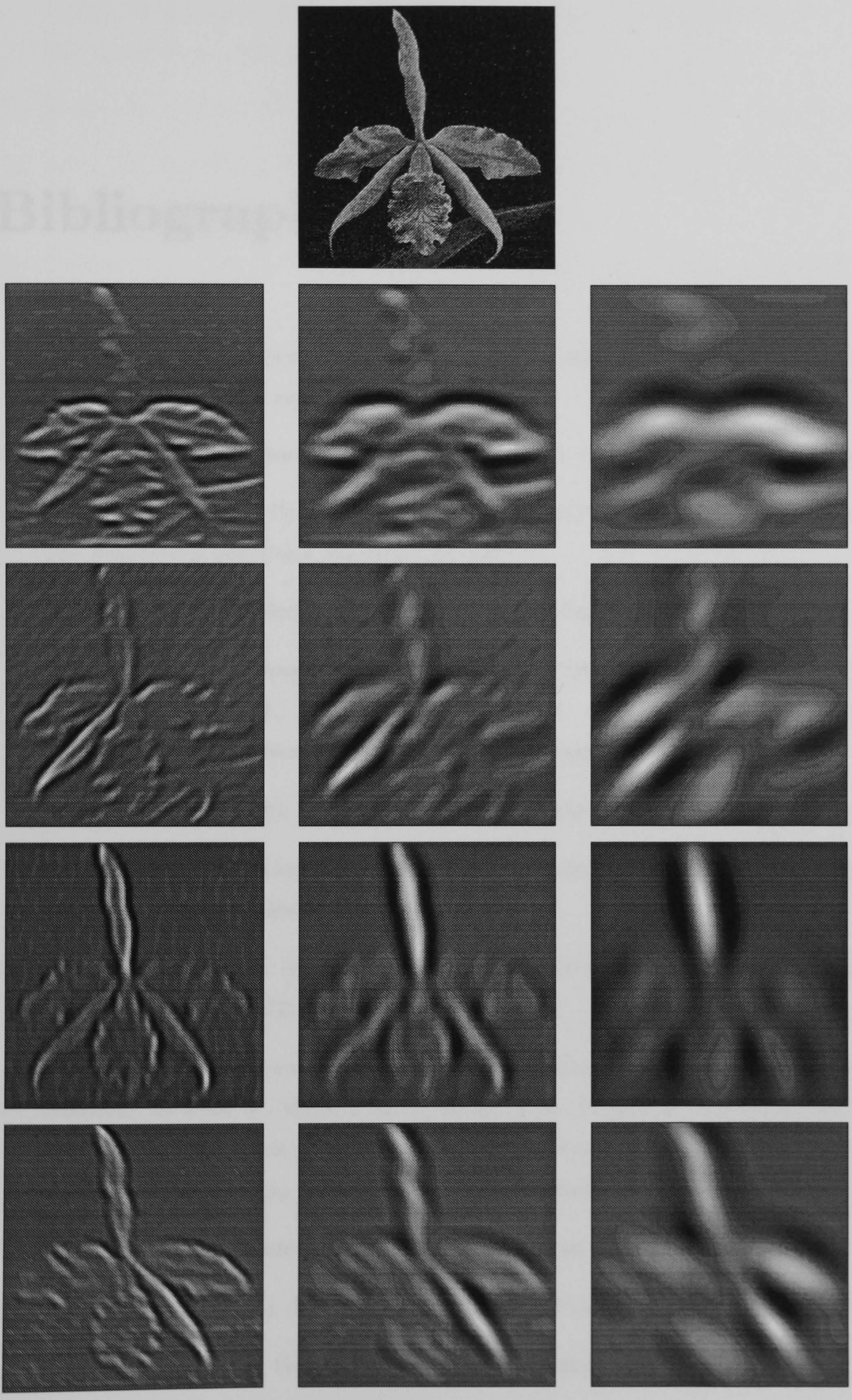


Figure 8.1: A 4-D array obtained by convolving the image at the top with scale and orientation specific filters. The scale of filters varies along the columns and the orientation varies down the rows (0, 45, 90 and 135 degrees).

Bibliography

[1] Arbib, M. A. (1987). Levels of modelling of mechanisms of visually guided behaviour. *The Behavioural and Brain sciences,* **10,** 407–465.

[2] Baker, R. R. (1981). *Human navigation: and the sixth sense.* Hodder and Stoughton, London.

[3] Ballard, D. H. (1986). Cortical connections and parallel processing: Structure and function. *The Behavioural and Brain Sciences* 9, 67–120

[4] Barlow, H. B. and Mollon, J. D. (1982). *The senses* Cambridge University Press.

[5] Barlow Jr., R. B., Powers, M. K., and Kass, L. (1985). Vision and mating behaviour in Limulus. In, Atema, J., Fay, R. R., Popper, A. N., and Tavolga, W. N. (Eds), *Sensory Biology of Aquatic Animals.* pp 419–434. Springer-Verlag, New York.

[6] Barlow Jr., R. B. (1990). What the Brain Tells the Eye. *Scientific American,* April 1990.

[7] Ballard, D.H. (1986). Cortical connections and parallel processing: Structure and function. *The Behavioural and Brain Sciences,* **9,** 67–120.

[8] Beer, R. D. (1995). A dynamical sysytems perspective on agent-environment interaction. *Artificial Intelligence,* **72,** 173–215.

[9] Beer, R. D. (1996). Toward the evolution of dynamical neural networks for minimally cognitive behaviour. In Maes, P., Mataric, M. J., Meyer, J. A., Pollack, J., and Wilson, S. W. (Eds.), *From animals to animats 4: Proceedings of the Fourth International conference on Simulation of Adaptive Behviour,* pp. 421–429. MIT Press Bradford Books, Cambridge MA.

[10] Berg, H.C. (1993). *Random walks in biology.* Princeton Univ Press, Princeton

[11] Bracewell, R. N. (1965). *The Fourier Transform and its Applications.* McGraw-Hill.

[12] Braitenberg, V. (1984) *Vehicles: Experiments in Syntheitc Psychology.* The MIT Press, Cambridge Mass.

[13] Brooks, R. A. (1986). *Acheiving Artificial Intelligence Through Building Robots.* Artificial Intelligence Memo, MIT.

[14] Brooks, R. A. (1991). Intelligence without representation. *Artificial Intelligence*, 47:139-159

[15] Bruce, V., and Green, P. (1985). *Visual Perception: Physiology, Psychology and Ecology.* Lawrence Erlbaum Associates.

[16] Campbell, N.A. (1993). *Biology.* Third Edition. Benjamin/Cummings, Redwood City, California.

[17] Cartwright, B. A. and Collett, T. S. (1983). Landmark Learning in Bees. *Journal of Comparitive Physiology A*, **151**, 521–543.

[18] Clark, A. & Thornton, C. (1994). Trading Spaces: Computation, Representation and the Limits of Uninformed Learing. *Cognitive and Computing Sciences, University of Sussex, Technical Report*

[19] Cliff, D. (1990). Computational neuroethology: A provisional manifesto. In, Meyer, J. A. and Wilson, S. W. (Eds), *From animals to animats: Proceedings of the first international conference on simulaion of adaptive behaviour (SAB90)*, pp 29–33. MIT Press Bradford Books, Cambridge Mass.

[20] Cliff, D. (1995). Computational neuroethology. In Arbib, M. A. (Ed), *The handbook of brain theory and neural networks.*, pp 626–630. MIT Press, Cambridge Mass.

[21] Cliff, D., Harvey, I., & Husbands, P. (1993). Explorations in evolutionary robotics. *Adaptive Behaviour* 2(1): 71–108

[22] Cliff, D., Harvey, I., & Husbands, P. (1997). Artificial evolution of visual control systems for robots. In, Srinivasan, M. & Venkatesh, S. (Eds), *From Living Eyes to Seeing Machines.* Oxford University Press.

[23] Collett, T.S. (1992). Landmark Learning and Guidance in Insects. *Phil. Trans. R. Soc. Lond B* 337, 297–303

[24] Collett, T. S., Cartwright, B. A., and Smith, B. A. (1986). Landmark learning and visiospatial memory in gerbils. *Journal of Comparitive Physiology A*, **158**, 835–851.

[25] Collett, T. S. & Baron, J. (1994). Biological compasses and the coordinate frame of landmark memories in honeybees. *Nature* **368**, 137–140.

[26] Dill, M., Wolf, R., & Heisenberg, M. (1993). Visual pattern recognition in Drosophilia involves retinotopic matching. *Nature* **365** 751–753

[27] Dusenbery, D. B. (1992). *Sensory Ecology: How organisms acquire and respond to information.* W.H.Freeman and company, New York.

[28] Ewert, J. P. (1984). Tectal mechanisms that underlie prey-catching and avoidance behaviours in toads. In H. Vanagas (Ed.), *Comparitive neurology of the optic tectum.* New York: Plenum.

[29] Ewert, J. P. (1987). Neuroethology of releasing mechanisms: Prey-catching in toads. *Behavioural and Brain sciences,* **10,** 337–405.

[30] Floreano, D., & Mondada, F. (1994). Automatic Creation of an Autonomous Agent: Genetic Evolution of a Neural-Network Driven Robot. In Cliff, D., Husbands, P., Meyer, A.J., and Wilson, S. W., Eds, *From animals to animats 3: Proceedings of the third international conference on simulation of adaptive behaviour (SAB94).*, pp 73–81. MIT Press, Cambridge Mass.

[31] Floreano, D., & Mondada, F. (1996a). Evolution of Homing Navigation in a Real Mobile Robot. *IEEE Transactions on Systems, Man, and Cybernetics–Part B: Cybernetics* 26(3).

[32] Floreano, D., & Mondada, F. (1996b). Evolution of Plastic Neurocontrollers for Situated Agents. In, P. Maes, M. Mataric, J-A. Meyer, J. Pollack, and S. Wilson. (Eds.), *From Animals to Animats IV,* Cambridge, MA: MIT Press, 1996.

[33] Franceschini, N., Pichon, J.M., and Blanes, C. (1992). From insect vision to robot vision. *Philosophical transactions of the royal society of London. B.,* **337,** 283–294.

[34] Fukushima, K. (1989). Analysis of the Process of Visual Pattern Recognition by the Neocognitron. *Neural Networks,* **2,** 413–420.

[35] Gallistel, C. R. (1990). *The Organization of Learning.* MIT Press, Cambridge Mass.

[36] Gallistel, C. R. (1995). The replacement of General-Purpose Theories with Adaptive Specializations. In Gazzaniga, M. S. (Ed), *The Cognitive Neurosciences,* pp 1255–1267. MIT Press, Cambridge Mass.

[37] Georgopoulos, A. P. (1995). Motor Cortex and Cognitive Processing. In Gazzaniga, M. S. (Ed), *The Cognitive Neurosciences,* pp 1255–1267. MIT Press, Cambridge Mass.

[38] Giurfa, M., Eichmann, B. and Meanzel, R. (1996). Symmetry perception in an insect. *Nature,* **133,** 458–461.

[39] Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison Wesley.

[40] Haykin, S. (1994). *Neural Networks.* Prentice Hall.

[41] Holland, J. (1975) *Adaption in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, USA.

[42] Hoyle, G. (1984). The scope of neuroethology. *The Behavioural and Brain Sciences.* **7**, 367–412.

[43] Hubel, D. H., and Wiesel, T. N. (1968) Receptive fields and functional architecture of monkey striate cotex. *Journal of physiology,* **195,** 215–243.

[44] Laughlin, S. B. (1987). Form and function in retinal processing. *Trends in neuroscience.* **10,** 478–483.

[45] LeCun, Y. and Bengio, Y., Pattern Recognition and Neural Networks. In Arbib, M. A. (Ed), *The handbook of brain theory and neural networks.*, pp 626–630. MIT Press, Cambridge Mass.

[46] Lin, L. (1992). Self-improving reactive agents based on reinforcement learning,planning and teaching. *Machine Learning* **8,** 293–321.

[47] Marevec, H. (1988) *Mind Children.* Harvard.

[48] Marr, D. (1977). Artificial intelligence—A personal view. *Artificial Intelligence,* **9,** 37–44.

[49] Marr, D. (1982). *Vision* New York: W.H.Freeman and Company

[50] McNaughton, B. L., Knierim, J. J., and Wilson, M. A. (1995). Vector Encoding and the Vestibular Foundations of Spatial Cognition: Neurophysiological and Computational Mecahnisims. In Gazzaniga, M. S. (Ed), *The Cognitive Neurosciences,* pp 1255–1267. MIT Press, Cambridge Mass.

[51] O'Carroll, D. (1993). Feature detectinf neurons in dragonflies. *Nature,* **362,** 541–543.

[52] Osorio, D., Getz, W. M., and Rybak, J. (1994). Insect vision and olfaction: different architectures for different kinds of sensory signal? In Cliff, D., Husbands, P., Meyer, A.J., and Wilson, S. W., Eds, *From animals to animats 3: Proceedings of the third international conference on simulation of adaptive behaviour (SAB94).*, pp 73–81. MIT Press, Cambridge Mass.

[53] Prescott, T., and Mayhew, J.,(1992). Building Long-Range Cognitive Maps using Local Landmarks. In Meyer, J.-A., Roitblat, H., and Wilson, S. (eds) *From Animals to Animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behavior* MIT Press Bradford Books, pp. 233-242.

[54] Redish, A. D. and Touretzky, D. S. (1997). Navigating with landmarks: computing goal locations from place codes. In K. Ikeuchi and M. Veloso, (eds.), *Symbolic Visual Learning.* Oxford University Press.

[55] Rumelhart, D. E., Hinton, G.E. & Williams, R.J. (1986). Learning internal representaions by error propagation. In: *Parallel distributedprocessing. Explorations in the microstructure of cognition. Vol 1.* Bradford Books/MIT press, Cambridge Mass.

[56] Snippe, H. (1991). *Human Perception of Spatial and Temporal Luminance Structure.* PhD Thesis, University of Utrecht

[57] Srinivasan, M. V. (1994). Pattern recognition in the honeybee: recent progress. *J. Insect Physiol.*, **3,** 183–194.

[58] Srinivasan, M. V., Zhang, S. W., & Rolfe, B. (1993) Is pattern vision in insects mediated by 'cortical' processing. *Nature.* **362** 539–540.

[59] Srinivasan, M. V., Zhang, S. W., Lehrer, M. and Collett, T. S. (1996). Honeybee navigation en route to the goal: visual flight control and odometetry. *Journal of Experimental Biology,* **199,** 237–244.

[60] Strausfeld, N. J., (1976). *Atlas of an insect brain.* Springer, Berlin.

[61] Sutton, R.S. (1988) Learning to predict by the methods of temporal difference. *Machine Learning* **3,** 9–44.

[62] Tesauro, G. (1992) Practical Issues in Temporal Difference Learning. *Machine Learning* **8,** 257–277

[63] Tinbergen, N. (1963) On aims and methods of ethology. *Z. Tierpsychol.*, **20,** 410–433

[64] Touretzky, D.S. & Redish, A. D. (1995). Landmark Arrays and the Hippocampal Cognitive Map. In: Niklasson, L. & Boden, M. (Eds), *Current Trends in Connectionism -Proceedings of the 1995 Swedish Conference on Connectionism* Hillsdale, NJ: Erlbaum.

[65] Touretzky, D. S., Wan, H. S., and Redish, A. D. (1994). Neural representations of space in rats and robots, In J. M. Zurada, R. J. Marks II, and C. J. Robinson (eds.), *Computational Intelligence: Imitating Life,* pp. 57–68. IEEE Press, Piscataway, NJ.

[66] Treves, A., Miglino, O., and Parisi, D. (1992). Rats, nets, maps and the emergence of places cells. *Psychobiology,* **20 (1),** 1–8.

[67] Walter, W. G., (1953). *The Living Brain.* Duckworth; reprinted by Pelican/Penguin, 1961, London.

[68] Wan, H. S., Touretzky, D. S., & Redish, A. D. (1994a). Towards a Computational Theory of Rat Navigation, In M. Mozer, P. Smolensky, D. Touretzky, J. Elman, and A. Weigerd, eds,.

Proceedings of the 1993 Connectionist Models Summer School, pp. 11–19. Lawrence Earlbaum Associates.

[69] Wan, H. S., Touretzky, D.S., & Redish, A. D. (1994b). Computing Goal Locations from Place Codes. In, *Proceedings of the 16th annual conference of the Cognitive Science society*, pp. 922–927. Lawrence Earlbaum Associates.

[70] Watkins, C.J.C.H. (1989) *Learning from delayed rewards.* PhD thesis, Kings college, Cambridge.

[71] Watt, R. J. (1988) *Visual Processing: computational, psychophysical and cognitive research.* Lawrence Erlbaum associates, Hillside, NJ.

[72] Wells, M. (1962). *Brain and behaviour in Cephalopods.* Heinman.

[73] Wilson, S. W., (1990). The animat path to AI. In, Meyer, J. A. and Wilson, S. W. (Eds), *From animals to animats: Proceedings of the first international conference on simulaion of adaptive behaviour (SAB90)*, pp 29–33. MIT Press Bradford Books, Cambridge Mass.

[74] Young, D. (1989). *Nerve Cells and Animal Behaviour.* Cambridge University Press, Cambridge.

[75] Zhang, S.W. & Horridge, G. A. (1992). Pattern recegnition in bees: size of regions in spatial layout. *Philosophical transactions of the royal society of London. B.,* **337,** 65–71.

[76] Zhang, S.W., & Srinivasan, M. V., & Horridge, G. A. (1992). Pattern recegnition in honeybees: local and global analysis. *Proceedings of the Royal society of London, B.* **248,** 55–61.