

This is to certify that the

thesis entitled

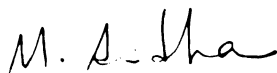
**Selective Visual Attention In
A Search Task: A Reinforcement
Learning Model**

presented by

Silviu D. Minut

has been accepted towards fulfillment
of the requirements for

M.S. degree in Comp. Sci. & Eng.



Major professor

Date 12/01/00

LIBRARY
Michigan State
University

PLACE IN RETURN BOX to remove this checkout from your record.
TO AVOID FINES return on or before date due.
MAY BE RECALLED with earlier due date if requested.

228 DATE DUE NOV 17 2002	DATE DUE	DATE DUE

SELECTIVE VISUAL ATTENTION IN A SEARCH TASK:
A REINFORCEMENT LEARNING MODEL

By

Silviu D. Minut

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Department of Computer Science

2000

ABSTRACT

SELECTIVE VISUAL ATTENTION IN A SEARCH TASK: A REINFORCEMENT LEARNING MODEL

By

Silviu D. Minut

This thesis proposes a model of selective attention for visual search tasks, based on a general framework for sequential decision-making. The model is implemented using a fixed pan-tilt-zoom camera in a visually cluttered lab environment, which samples the environment at discrete time steps. The system is a learning agent which has to decide where to fixate next, based purely on visual information, in order to reach the region where a target object is most likely to be found. The model consists of two interacting modules. A reinforcement learning module learns a policy on a set of regions in the room for reaching the target object, using as objective function the expected value of the sum of discounted rewards. By selecting an appropriate gaze direction at each step, this module provides top-down control in the selection of the next fixation point. The second module performs “within fixation” processing, based exclusively on visual information. Its purpose is twofold: to provide the agent with a set of locations of interest in the current image, and to perform the detection and identification of the target object.

To My Family

ACKNOWLEDGMENTS

First and foremost I would like to thank my advisor, Dr. Sridhar Mahadevan. Numerous discussions ranging from machine learning and reinforcement learning, to human and computer vision, pattern recognition and cognitive science, lead to ideas that shaped this research. I would also like to thank the other members of the committee, Dr. George Stockman and Dr. John M. Henderson for their support and for their invaluable expertise in computer vision and in vision cognition. I am also grateful to Dr. John Weng and Dr. Fred Dyer for their insight and suggestions. Last, but not least, I would like to thank the members of the SIGMA group at Michigan State, for all the ideas that we have exchanged, and in particular to Richard Falk, for providing some human data from the scan pattern experiments.

TABLE OF CONTENTS

LIST OF TABLES	vii
LIST OF FIGURES	viii
1 Introduction	1
2 Background	5
2.1 Biological Motivation	5
2.1.1 Saliency Map Theory	12
2.2 Reinforcement Learning	15
3 Problem Definition	21
3.1 Research Questions and Task Definition	21
4 Related Work	25
4.1 A Saliency Map Implementation	25
4.2 Reinforcement Learning	27
5 Model Description	31
5.1 The System Overview	31
5.2 The Q-learning Module	34
5.2.1 Feature Extraction	34
5.2.2 States, Actions and Reward	39
5.3 The Vision Module	42
5.3.1 The Symmetry Operator	43
5.3.2 Histogram Back-projection	45
5.3.3 Histogram Intersection	48
5.4 More Visual Routines	50
5.5 Algorithm	56
6 Experimental Results	63
6.1 Learning a Policy for Finding the Target	63
6.2 Executing the Learned Policy	67
6.3 Re-training for a Different Target	68
6.4 Comparison with Random Search	70
6.5 Comparison with Exhaustive Search	71
7 Discussion	73

LIST OF TABLES

6.1	Average number of fixations for the three experiments starting from two test points. In experiment 1 testing was done every 5-th epoch. The rest of the epochs were training epochs. Experiment 2 consisted only in executing a learned policy, with the target object inverted and slightly displaced. In experiment 3, a random search was done.	72
-----	--	----

LIST OF FIGURES

2.1	Human fixations (the red dots) in an indoor scene. Most of the fixations fall on objects. There are almost no fixations on large, flat, uniform surfaces (floor, walls). Courtesy of the MSU Eye-Lab. Images in this thesis are presented in color.	10
2.2	Human fixations (red dots) in an outdoor scene. Courtesy of the MSU Eye-Lab.	11
3.1	(a) RGB image of the target object. (b) Panoramic image of the environment. Images in this thesis are presented in color.	23
5.1	Overall architecture of the system. The top module performs the feature extraction necessary to cluster the images online. The resulting regions are used as states in a Q-learning program and a policy for moving towards the target region is learned. The bottom module implements the “within fixation” visual processing, consisting of computation of interesting regions in the image and object recognition.	32
5.2	(a) Low resolution template of the target object. (b) High resolution template.	33
5.3	Kullback distance of pairs of similar images (red) and on different images (green)	34
5.4	Top: feature vectors (4 concatenated color histograms) for two similar images (bottom)	35
5.5	Top: feature vectors extracted from two different images (bottom)	36
5.6	The shaded region represents a sector in image A. The most salient point is computed (the blue square) and a new image B, centered at the new location is taken.	40
5.7	If the reward is found in the periphery (image A) then upon subsequent visits to the same region (image B) the reward could be missed, despite the fact that the agent executed the “right” action.	41
5.8	Two edge pixels p_i and p_j will vote for their midpoint. The strength of the vote depends on the distance between p_i and p_j , and on the length and the orientation of the intensity gradient at each of the two pixels.	43
5.9	(a) Sample image. (b) Symmetry map obtained by maximizing the vote for pixels at distance $\mu = 130$	45
5.10	(a) Sample search image. (b) Model (target) image. (c) Histogram back-projection of (b) onto (a).	46

5.11	(a) Indoor image. The dots represent the recorded human fixation points. Courtesy of the Michigan State University EyeLab. (b) Intensity image of (a). The red squares are the cells with large variance.	51
5.12	(a) Intensity map for the image in Figure 5.11 (a). (b) Contrast map. The brighter regions are the most salient. Compare with the human fixations in Figure 5.11 (a).	53
5.13	Gaussians used to model color opponency. The red curve weighs the responses of the central red cones, while the green curve weighs the (inhibitory) responses of the surrounding green cones. The blue curve is the difference of the two gaussians.	55
5.14	(a) Original RGB image. (b) Red-green opponency map. (c) Green-red opponency map. (d) Red-green opponency using (b) and (c) as the red and green channels. (e) Green-red opponency using (b) and (c) as the red and green channels. The brightest regions are the ones in which the center-surround opponency is the largest. The green-red opponency picks up the flag and the computer chair, while the red-green opponency picks up the red box.	55
5.15	Red: distribution of high resolution images containing the target object. Green: distribution of high resolution images not containing the target object.	58
5.16	Left: Low resolution image. The points p_1, p_2, p_3 are produced using the histogram back-projection of the low resolution model M_{low} . At each p_i , a score s_i is computed using histogram intersection. Right: High resolution image centered at p_3 . The high resolution model M_{high} is first back-projected onto the image, and then matched using histogram intersection. If the target was not found at any of the p_i s then the camera returns to the pan and tilt coordinates it had before the local search, and the threshold T is set to $\min s_i$	59
6.1	Average number of fixations for every 5-th epoch over five trials of 400 epochs each. At every 5-th epoch, the policy learned was tested starting from pan=500, tilt=-100.	64
6.2	Average number of fixations for every 5-th epoch over five trials of 400 epochs each. At every 5-th epoch, the policy learned was tested starting from pan=200, tilt=0.	64
6.3	Initial scan path (green) and learned scan path (red). The rectangle $[-860, 860] \times [-290, 290]$ represents the actual pan-tilt range of the camera. The search starts at (300, 50) and the target object is found around (-380, 30).	65
6.4	The sequence of fixations corresponding to the learned path (red) in Figure 6.3. The camera starts from region (1) and gradually moves towards the goal region (5). Here, a sufficiently good candidate is found at low resolution (1), the camera zooms in (6) and does a high resolution match. This sequence of images corresponds to the red scan path in Figure 6.3	66
6.5	States learned (the blue squares) and the best actions. Action A_0 is represented by a red square around a state. Actions $A_1 - A_8$ are represented by a red segment originating at the state. The target is the green square.	66

6.6	Number of saccades per epoch in finding the object inverted, displaced by 60 cm from the training position. Testing from pan=500, tilt=-100.	67
6.7	Number of saccades per epoch in finding the object inverted, displaced by 60cm from the training position. Testing from pan=200, tilt=0.	68
6.8	(a) The target object in the third experiment was the green flag. (b) Low resolution template. (c) High resolution template.	69
6.9	Average number of fixations for finding the MSU flag (Figure 6.8) for every 5-th epoch over five trials of 400 epochs each. At every 5-th epoch, the policy learned was tested starting from pan=500, tilt=-100.	69
6.10	Average number of fixations for finding the MSU flag (Figure 6.8) for every 5-th epoch over five trials of 400 epochs each. At every 5-th epoch, the policy learned was tested starting from pan=200, tilt=0.	70
6.11	Number of saccades per epoch in a random search. Testing from pan=500, tilt=-100.	71
6.12	Number of saccades per epoch in a random search. Testing from pan=200, tilt=0.	72

Chapter 1

Introduction

The problem of *visual search* is to find an object in a large, usually cluttered environment (e.g. a pen on a desk) [52]. In solving such a problem it is preferable to use wide field of view images, in order to analyse as much as possible of the environment under scrutiny. On the other hand, small objects require high resolution images, which in combination with the wide field of view requirement leads to a very high dimensional input array. *Foveated vision* is nature's method of choice in solving this problem and is a dominant characteristic of the vision system of virtually any vertebrate species with well developed eyes [3]. It consists of processing of the visual input at non-uniform resolution, due to an increased density of photo-receptors on a small central patch (the *fovea*) at the level of the retina. As such, foveated vision achieves the compromise between the need for a wide field of view and the need for high acuity.

Foveal image processing reduces the dimension of the input data, but in turn generates an additional sequential decision problem. Choosing the next fixation point

requires an efficient gaze control mechanism in order to check salient objects, to determine whether they are the target or not.

From an engineering standpoint, a sequential attention mechanism is attractive because it has the potential of requiring only sparse local models [4]. However, the visual attention mechanism raises a plethora of difficult questions. In the first place, since the next fixation point is generally not in the fovea, its selection must be done based on coarse, low resolution visual information, without a thorough understanding of its semantics. The question is then, what low level features are necessary in order to decide what to attend to in the next fixation. Koch and Ullman [22] propose a saliency map theory which is a task independent, bottom-up model of visual attention. In this framework, Itti and Koch [16], extract three types of feature maps (a color map, an edge map and an intensity map) and fuse them together in a unique map. However, the selection of the next fixation must require some top-down control since low-level visual information is usually not sufficient. Hence the second major question is how to implement a high level, top-down mechanism to control the low level, reactive attention? Tsotsos et. al. [46] propose a model of visual attention which tries to selectively tune visual processing by means of a top-down hierarchy of winner-take-all processes. Finally, since the vision system samples the environment, some information must be retained from one fixation to the next, and integrated across saccades, to produce a global understanding of the scene. The nature of this information is the third unknown of major concern. We propose an overall model that integrates top-down gaze control with bottom-up reactive saliency map processing, based on reinforcement learning.

In the remainder of this section we present an outline of the thesis.

In Chapter 2 we introduce the relevant terminology and present some background and motivation from a cognitive science perspective, as well as give a brief introduction in reinforcement learning. Some fundamental issues pertaining to visual attention are outlined.

In light of the problems raised in Chapter 2, we formulate in Chapter 3 the questions that are addressed in this thesis and we state in mathematical terms the concrete task being solved.

Chapter 4 is devoted to related work, where a number of representative previously proposed models of visual attention are discussed.

The core of this thesis is described in Chapters 5 and 6. Section 5.1 provides an overview of the model and explains the logical decomposition of our system into two modules: a reinforcement learning module, which provides the top-down control in the search task, and a vision module, consisting of bottom-up visual routines that implement our version of a saliency map and a recognizer used to identify the target object of the search. We proceed to describe the components of each module in the subsequent sections, independently of one another. The feature extraction and the Q-learning routines are described in sections 5.2, 5.2.1 and 5.2.2, while the visual routines are detailed in Section 5.3. The individual components are then “assembled” together in Section 5.5, where a detailed description of the algorithm is given.

The performance of the search system is evaluated in Chapter 6, where four experiments are described. The results show that (1) the number of fixations to the goal decreases during training, (2) after training, the system can find the target object in

a slightly different location within one region of the environment, (3) the agent can be re-trained to find another object in the same environment without fine tuning any parameters. Finally, a comparison with a random search agent and with an exhaustive search agent is done, indicating that the learning agent performs better in both cases.

Chapter 7 summarizes the strengths of the system, as well as its shortcomings. Some technical issues regarding the implementation are presented, and some possible improvements to feature extraction. Finally, Chapter 8 outlines a few promising directions for further work.

Chapter 2

Background

2.1 Biological Motivation

Broadly speaking, computer vision is a research area with the practical goal of building a machine that (understands what it) sees. In doing so, researchers often find their inspiration in biological systems, and often propose computational models that try to explain or approximate the observed biological evidence. One of the first comprehensive theories of computer vision was formulated by David Marr [26]. It was the dominant paradigm in computer vision of the 80's and it continues to be a major one even today. The aim of computer vision, according to this theory, is to recover the 3D structure of the external world, from 2D images, in an attempt to form a stable, detailed, spatiotopic representation of the world. Attempts to implement such a system failed to perform well in realistic environments. A few researchers have used psychophysical evidence to argue for new biologically inspired approaches in computer vision [4]. We shall describe some of this evidence and its implications to vision, but

in order to do so, we shall briefly introduce some terminology first.

It is estimated that more than half of the neurons in the brain of the macaque monkey are devoted to vision (though fewer in the human brain) [11], [32]. This is indicative of the importance and the complexity of vision. The brain must analyse the sensory input, then synthesize the information in an attempt to “make sense” of the world, and act in an appropriate, optimal way. However, given that the human field of view is about 180° , it is a highly nontrivial task to process and interpret all the visual information at constant, high resolution, even for a massively parallel super-computer such as the brain, for dimensionality reasons. It is also not necessary to keep *detailed* information about every single point in the visual field.

The biological solution starts with the design of the eye and consists in a trade-off between high visual acuity and amount of processing. The *fovea* is anatomically defined as a small, central region on the retina, with a very high density of receptive cells (cones). The density of the photo-receptors (and with it the visual acuity as well) decreases exponentially from the fovea towards the periphery. To compensate for any potential loss of information incurred by the decrease in resolution in the periphery, the eyes are rapidly re-oriented via very fast (up to $900^\circ/s$) ballistic motions called *saccades*. The length of the saccade varies with the task, but in general, the order of magnitude is a few degrees (see [35]). *Fixations* are the periods between saccades (about 3 per second) during which the eyes remain relatively fixed, the visual information is processed and the location of the next fixation point is selected. In humans, the fovea has a diameter of about 1.5 mm, and accounts for about 2° of the field of view [32] [35].

This dimensionality reduction leads to a sequential decision problem: rather than processing all the visual field at once at constant high resolution to produce a detailed global picture of the whole environment, the visual system devotes a substantial part of its resources to the processing of a limited region of high resolution, in conjunction with an efficient attention mechanism, which selects the next fixation point in such a way that the brain appears to get just the right amount of information necessary for the task at hand.

In the early 90's two closely related paradigms started to affect the course of vision research in computer science: active vision [2] and animate vision [4]. In essence, the active vision paradigm proposes that instead of trying to recover the world in the form of a task independent, detailed internal representation, one should try to find task specific solutions. These solutions could be found by decomposing the particular task at hand into simpler sub-tasks, with simpler *qualitative* solutions. As Aloimonos points out [2], an algorithm that requires accuracy of a few decimals cannot be robust under the virtually infinite number of environments, hence the need for qualitative algorithms.

Animate vision also acknowledges the fact that it is not necessary to create rich 3D representations of the world. An animate vision system ideally would be endowed with anthropomorphic features, such as binocular, foveated vision and a high speed gaze control mechanism. Animate vision systems can take various actions (e.g. move, or zoom in/out) in order to optimize or simplify visual search. The computations are carried out in an exocentric coordinate system (as opposed to an egocentric coordinate system in Marr's paradigm), which has the advantage of being independent of the

motion of the observer (robot). The camera is not fixed, but can make approximate motions to acquire new information. Understanding vision as a continuous process of mapping perception to action, allows for (reinforcement) learning algorithms which can provide a vision system with more sophisticated behavior.

O'Regan [30] suggests that only minimal information about the scene is represented internally. The scene itself could serve as "external" memory. In a now classical experiment, Ballard [5] confirmed O'Regan's finding. Specifically, human subjects were asked to reproduce a configuration of colored blocks displayed on a computer monitor. The display was divided into 3 regions: the model region, in which the given configuration was shown, the source, which contained an unordered set of colored blocks, and the workspace, which was the area where the copy of the model was supposed to be assembled. Had the subjects built a detailed internal representation of the model in their mind first (as suggested by the Marr paradigm), one (possibly long) look at the model would have been enough to produce a copy in the working area. Without exception though, the subjects made saccades back and forth between the model and the workspace, accomplishing the task by selecting one block at the time. This suggests that it is more economical to retain only the necessary information locally both in space and time, and sample the environment as necessary, rather than keep an internal representation of the scene.

Another piece of evidence against computing and storing a stable internal model of the world is the *change blindness* effect [38], [37]. It has been observed that even significant (sudden) changes in a scene go undetected if they occur during a saccade ¹.

¹It is known that visual input during a saccade is essentially turned off, a phenomenon termed

Again, had the brain maintained a detailed spatio-temporal model of the scene, the changes would have been noticed.

Selective attention is the common feature of biological systems, but it raises a number of questions, some of which we shall discuss. Rensink [37] points out that there are two categories of problems. One category addresses questions about retinotopic representations and processes that occur within a single fixation. Equally important is what happens from fixation to fixation in terms of how and what kind of information is integrated across saccades, as well as how this information is coordinated with the task at hand. We shall point out some fundamental questions from each category. First, the “within fixation” questions.

Yarbus [54] and subsequently several other studies (see [35] and the references therein), showed by tracing the fixation points in a scene, that the human scan patterns vary with the task. Figures 2.1 and 2.2 shows human scan patterns recorded with an eye-tracker at the MSU EyeLab. Despite these variations, the fixation traces have some common characteristics. It is noteworthy, for instance, that people almost *always* fixate on objects, both in indoor images and in outdoor images, and almost never on flat, uniform, empty spaces (e.g. floors, walls). It has been postulated [49] that there are (at least) two fundamental operations that a visual system must be able to perform: first it must be able to tell “*where*” in the image the interesting regions are, and then, once attention is devoted to one of these regions, it must tell “*what*” that region represents. In general, coarse, peripheral information is not sufficient for object recognition. For instance, surprising as it may seem, color perception

saccadic suppression [35].

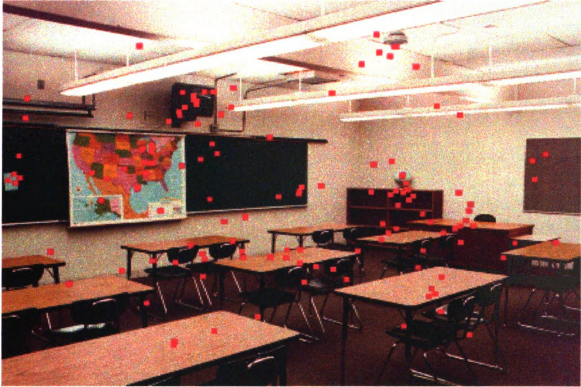


Figure 2.1: Human fixations (the red dots) in an indoor scene. Most of the fixations fall on objects. There are almost no fixations on large, flat, uniform surfaces (floor, walls). Courtesy of the MSU Eye-Lab. Images in this thesis are presented in color.

is heavily depreciated beyond 20-30 degrees off the optical axis, due to the decrease in cone density [32]. The peripheral information is used mainly to select the next fixation point, after which recognition occurs based primarily on foveal information. In light of these facts, a logical paradox occurs: how does the visual system decide where something *interesting* is in the scene, *before* knowing what it is? Trying to answer the “what” question first seems equally hopeless, for how can it be decided what an object is, before looking at it (i.e. before knowing where it is)? One theory that attempts to explain the selection mechanism is the *saliency map* framework, which we discuss in some detail in the next section.

One fundamental problem in gaze control, part of “across fixations” research is



Figure 2.2: Human fixations (red dots) in an outdoor scene. Courtesy of the MSU Eye-Lab.

how integration across saccades is performed, i.e. what information the brain retains from fixation to fixation in order to produce the illusion of a coherent, uniform resolution image, rather than a collection of snapshots of various regions in a scene. It has been suggested (see [35] and references therein) that information from successive fixations is stored in a high capacity visual buffer. The main technique used in vision research to investigate this problem is to present human viewers with an object extrafoveally, and then instruct them to direct their gaze towards the object. During the saccade the object is replaced with another object which the subjects must name. Studies in object perception [35], however, show that the visual information is not stored in a point by point fashion, but rather as relatively abstract representations.

Integration of information across saccades remains an open problem [35].

2.1.1 Saliency Map Theory

If the understanding of a scene is a sequential process, then naturally, we must try to understand the mechanism that allows the shift of attention from one region to another.

An initial step is to observe that not all visual processes are created equal. Some features (e.g. luminance, direction of motion) can be processed quickly and in parallel over the entire field of view, whereas more cognitive processes such as shape analysis, or fine recognition, have much higher complexity and require an increased amount of resources, which necessarily must be carried out locally, on a restricted region called focus of attention which most of the time (but not always!), coincides with the fixation point. The former processes constitute the “pre-attentive stage”, whose role is to select a single region as the focus of attention, which will subsequently be analysed by the latter processes - the “attentive stage”.

There exists experimental evidence that strongly supports the “selective attention” theory. In the first place, experiments by Treisman [47], [48] showed that search for a target defined by a single visual feature (e.g. color, or edge orientation) takes place in parallel over a visual display, whereas targets defined by a conjunction of several features is done sequentially (e.g. find a vertical red line among many red and blue lines, vertical or horizontal). It has also been established [7], [20], [47] that only a limited set of features (e.g. color, edge orientation and curvature) can

be detected in parallel. Physiological evidence based on recordings of neuronal activity in awake behaving monkeys also supports the selective processing of the visual data [8], [12], [13].

In light of these facts, Koch and Ullman [22] proposed a model of visual attention called *saliency map*, aimed at explaining the shift of attention and the saccadic eye movements. We describe their model in the remainder of this section. The result of the processing in the pre-attentive stage is a set of topographical cortical maps (possibly at different spatial resolutions), which are based on low-level visual information (color, edge orientation, texture, contrast, disparity, direction of motion) and provides an early representation of the environment. Each of these feature maps codes for conspicuity within one feature: the more different a location is from its surrounding, the more conspicuous (i.e. more salient) it is. The feature maps are then combined into a saliency map by computing at each location a weighted sum of the saliency in each of the features. We must point out that it is nontrivial to compute such a weighted sum, because some of the features are inherently not comparable (e.g. it is not obvious how to compare color conspicuity with edge orientation). In fact, it is quite possible that these weights may vary, and may get top-down (semantic) influence from some higher cortical centers.

Once a salient location has been selected, more abstract properties of that region are created and a central, non-topographical representation of that region is formed. A “winner-take-all” network is proposed for finding the most salient region in the saliency map. A second network is also necessary for building the abstraction of the attended region into a central representation.

Since the saliency of a region in the field of view is based on low-level visual features, it is extrinsic to the visual system, i.e. it is entirely determined by the environment. Then, the most salient region in the early representation, remains the most salient also after that region has been attended. Naturally, then, the question is, if the maximum saliency region is always selected, how does one ever move the eyes to another location? Koch and Ullman suggest that higher cortical centers decrease the saliency of the attended region, so that after some time, that region is no longer the most salient in the field of view, and another maximum is selected elsewhere.

A few comments are in order here. In the first place, as we have already mentioned, it is not trivial to combine the various feature maps into a single saliency map. Secondly, in the saliency map framework, it is not clear when the high level processing occurs. The saliency map only tries to explain how the next focus of attention is chosen, and, as presented above, semantics are given little consideration. In the absence of semantics, the selection of the next fixation point is a reactive, deterministic process (once an environment is given), which does not require any voluntary, cognitive actions; in other words, it is a reflex. While this may be the case during the first few fixations when presented with a new scene, we believe that as the scene is being gradually understood, semantics, and the specific task at hand have an increased role in selecting the next fixation point.

We close this section by mentioning that while there is biological evidence for the existence of various feature maps, not all researchers accept the idea of a unique topographic map to represent the most salient stimuli [10].

2.2 Reinforcement Learning

We emphasized in the previous section the necessity of a top-down control mechanism to obtain the functionality of a gaze control system. Since we treat visual attention as a sequential decision problem, we shall use reinforcement learning techniques [43] to implement the top-down control in a task specific way. In this section we present the basic formalism of sequential decision-making.

An agent is a system which can be in one of a finite set of *states*, and which can take *actions*. Depending on the problem at hand, the information used to define the states could be *extrinsic* to the agent, describing properties of the environment (e.g. location information, “obstacle ahead”, “rainy day”, etc.), or *intrinsic*, internal to the agent (e.g. “hungry”, “happy”, “angry”, “sure”, “unsure”, etc.). In general, there is one set of actions for each state, but for simplicity, we shall assume that at each state the agent can choose from the same set of actions. The actions change the state of the system and the agent must re-estimate the state by observing the environment. The agent gets rewarded for each action it takes, and tries to learn what action is the best in each state, with respect to some optimality criterion, by moving from one state to another until some termination conditions are met.

Typically, the components of a sequential decision making problem are as follows:

- **States:** $S = \{s_1, s_2, \dots s_n\}$.
- **Actions:** $A = \{a_1, a_2, \dots a_m\}$.
- **Transition probabilities:** $P(s'|s, a)$ of reaching state s' given that the agent

was in state s and has taken action a . In general, the transition from s to s' by executing action a depends not only on s and a , but on the whole sequence $s_0, a_1, \dots, s_n = s$ up to the current state. In many sequential decision problems, however, the transition from a state to another does not depend on history, i.e.

$$P(s_{n+1}|a_{n+1}, s_n, \dots, a_1, s_0) = P(s_{n+1}|a_{n+1}, s_n) \quad (2.1)$$

Equation (2.1) is called the *Markov property* and systems in which it is satisfied, are called a *Markov Decision Processes* (MDP). We assume the Markov property throughout this thesis.

- **Reward:** $R(s, a, s')$ for transiting from state s to s' as a result of action a . The reward can be positive, zero, or negative (cost). We assume that rewards are bounded in absolute value. Since the action a puts the system in state s' with probability $P(s'|s, a)$, the quantity

$$r(s, a) = E(R(s, a, s')|s, a) \quad (2.2)$$

$$= \sum_{s'} P(s'|s, a) \cdot R(s, a, s') \quad (2.3)$$

represents the one step expected reward for executing action a in state s .

- **Objective function:** A function depending on long term reward, which the agent must optimize. For instance the agent might try to maximize the sum of

rewards:

$$r(s_0, a_1) + r(s_1, a_2) + \dots$$

where a_i is the action taken at moment i , s_i is the state at time step i and s_0 is the state the agent starts from.

To guarantee that the above series is finite, we assume the rewards are bounded and we discount each of the terms by a factor $0 < \gamma < 1$. We require the agent to maximize at each state the expected value of the sum of discounted rewards:

$$E\left(\sum_{t=1}^{\infty} \gamma^{t-1} \cdot r(s_{t-1}, a_t)\right) \quad (2.4)$$

See [21], [24] for more examples of optimality metrics.

A (stationary, deterministic) policy is a function $\pi : S \rightarrow A$. The agent follows a policy if in each state s it always chooses the same action $a = \pi(s)$. Given an optimality metric and a policy π , we can define for each state s a value $V^\pi(s)$. For instance, for the optimality metric (2.4), the value of state s is

$$V^\pi(s) = E\left(\sum_{t=1}^{\infty} \gamma^{t-1} \cdot r(s_{t-1}, \pi(s_{t-1})) \mid s, \pi\right) \quad (2.5)$$

$$= r(s, \pi(s)) + \gamma \cdot \sum_{s' \in S} P(s' \mid s, \pi(s)) V(s') \quad (2.6)$$

where $s_0 = s$ and $s_1 = s'$.

By “optimizing” the objective function we mean that the agent must find an optimal policy, i.e. a policy π^* such that

$$V^{\pi^*}(s) \geq V^{\pi}(s) \quad \forall s \in S$$

For simplicity we denote $V^* = V^{\pi^*}$. By (2.6) we have that

$$V^*(s) = \max_{a \in A} [r(s, a) + \gamma \cdot \sum_{s' \in S} P(s'|s, a) V^*(s')] \quad (2.7)$$

for any state s . Equations (2.7) are called the Bellman optimality equations, and there are two known dynamic programming algorithms that can be used to solve them: value iteration and policy iteration [21], [24], [33], [43]. These algorithms can be applied when we have a model of the MDP, i.e. when the transition probabilities and the reward functions are known *a priori*. This may be the case in some manufacturing problems, but it rarely happens in real robotics problems. Indeed, a robot does not know the outcome of an action until it has completed that action.

There exists, however, an algorithm called *Q-learning* due to Watkins [50] which can be used to solve the Bellman equations even when the transition probabilities or the rewards are not known. The algorithm is iterative, it is proven to converge, and finds a solution of the Bellman equations.

To begin with, suppose that V^* is a solution of (2.7). For each pair (s, a) define a function

$$Q^*(s, a) = r(s, a) + \gamma \cdot \sum_{s' \in S} P(s'|s, a) V^*(s') \quad (2.8)$$

which is precisely the quantity under max in Equation (2.7). Clearly then,

$$V^*(s) = \max_{a \in A} Q^*(s, a) \quad (2.9)$$

Substituting (2.9) back into (2.8) we get

$$Q^*(s, a) = r(s, a) + \gamma \cdot \sum_{s' \in S} P(s'|s, a) \max_{a' \in A} Q^*(s', a') \quad (2.10)$$

Unlike in the Bellman equations, in equations (2.10) the transition probabilities are not under max. The quantity

$$\sum_{s' \in S} P(s'|s, a) \max_{a' \in A} Q^*(s', a')$$

is the expected value of $\max_{a' \in A} Q(s', a')$. The Q-learning algorithm tries to find the function $Q^*(s, a)$ through successive approximations. Once Q^* is found, we can find an optimal policy by

$$\pi^*(s) = \arg \max_{a \in A} Q^*(s, a)$$

Let Q_n be the approximation at time step n . Q_0 is initialized randomly, or it is identically 0. The iterative step is given by the following update equation:

$$Q_{n+1}(s, a) = Q_n(s, a) + \alpha_n[(r(s, a) + \gamma \cdot \max_{a' \in A} Q_n(s', a')) - Q_n(s, a)] \quad (2.11)$$

where α_n is a learning rate parameter. The transition probabilities are not present in the update equation (2.11). It can be proven rigorously [50] that the sequence Q_n is convergent to a function Q^* which satisfies (2.10), provided that $\sum \alpha_n = \infty$ and $\sum \alpha_n^2$ is finite and provided that all (state, action) pairs are visited infinitely often.

Chapter 3

Problem Definition

3.1 Research Questions and Task Definition

Now that we have introduced the basic ideas in active vision and reinforcement learning, we can formulate more precisely the theoretical questions that we investigate in this thesis.

The first question is, how can perception and action be unified in a coherent, possibly task dependent visual behavior. In other words, given that the agent recognizes (with some certainty) an image, how does it decide what to do afterwards? We propose a model of visual attention which comprises a low level, reactive layer, which essentially implements a saliency map, controlled by a high level, semantic layer. The addition of the top layer goes beyond previous gaze control models, in which the emphasis was on selecting the most salient point in a task independent way.

Secondly, we investigate to what extent reinforcement learning can be used to

integrate information across saccades. We propose that only minimal visual information about a region in the environment is retained, together with the direction of the saccade from that region towards some goal. This is analogous to landmark navigation that organisms (from insects to humans) are known to perform.

Finally, we ask whether it is possible to learn relationships between objects. For instance, if the agent is searching for a pen, then, to restrict the search, we would like it to know from prior experience that a pen is likely to be found on a desk. Many examples from real life, show that people do use this kind of prior knowledge: we look for keys in certain places, we look for a book in a bookcase, etc. We believe that reinforcement learning can partially answer this question, but a more complete solution relies heavily on the ability of the agent to recognize *classes* of objects, rather than specific objects (e.g. the agent can recognize “books”, or “pens”, not just “this book”, or “this pen”).

A visual search for a relatively small object which is not necessarily in an exact location, but which can usually be found near some other larger fixed object in a cluttered environment, seems to have the necessary complexity to address the above issues. At any given moment the system can only process a limited region in the environment, at high resolution, and knowledge about each of the regions visited must be “remembered” from one visual frame to another. Also, moving to another region requires some way of finding a new fixation point, which requires solving the aspect of visual saliency. The fact that the target object is generally in one region of the room, requires that the saccades be *directed* consistently towards the target region, hence the problem of top-down control.

Formally, we define the search task as follows.

Task: Find a pre-specified object (Figure 3.1 (a)) in a fixed position in a room (Figure 3.1 (b)), using a fixed, pan-tilt-zoom camera.

Input: An RGB image of the target object T .

Output: A set $L = \{L_1, L_2, \dots, L_n\}$ of landmarks (regions) in the room, and a policy on L for finding the target T .

Performance Metric: Number of saccades the camera makes from an arbitrary (but fixed) position, until it finds the target T .

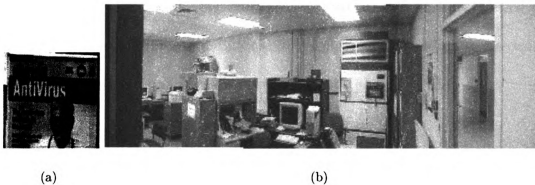


Figure 3.1: (a) RGB image of the target object. (b) Panoramic image of the environment. Images in this thesis are presented in color.

The general approach is to create the set L incrementally, by using an instance based approach of clustering the images. Each image is represented by means of color histograms. Q-learning is used to learn the policy on the set L . The next fixation point is chosen by using two feature maps (rather than a single saliency map): we do a low resolution search for a candidate location, based on color, followed by a

high resolution match at each candidate location. If the target is not found, then we compute symmetries in the image using an adapted version of the symmetry operator described in [36], and choose the maximum in a given direction as the next fixation point.

We want to emphasize here that although a gaze control system involves a great deal of image processing and pattern recognition techniques, we do not attempt to build a (nearly) perfect, general purpose recognizer, nor the perfect saliency map. What we strive for, is to introduce task dependent, top down control in visual attention. Improvements both in the image processing component, as well as in the reinforcement learning component will be added subsequently, based on the insight gained from the current work. For now, however, we see this work as the first step towards understanding the computational issues involved in building a gaze control system capable of operating in realistic environments.

Chapter 4

Related Work

4.1 A Saliency Map Implementation

A number of approaches to visual attention, visual search and gaze control have been proposed [1], [9], [16], [22], [29], [34], [39], [46], [53]. We shall succinctly describe some of them in this section.

Perhaps the most comprehensive implementation of the original saliency map theory proposed by Koch and Ullman was done by Itti and Koch [16], [15] who implemented a model of a task independent, pre-attentive selection mechanism. A saliency map is built using a bottom-up approach. To this end, three types of feature maps are built and combined into a unique map: a color map, an edge-orientation map and an intensity contrast map, each at different scales in a pyramidal scheme, yielding a total of 45 feature maps. Each of these features uses a center-surround structure that mimics the visual receptive fields, which is meant to pick up regions that are in sharp contrast with their surrounds. For the color contrast, for instance, the difference

between the red value at one pixel and the green value of surrounding pixels at a different scale is computed, resulting in a red-green contrast map. Similarly, blue-yellow contrast maps are computed and then combined with the red-green maps. The same center-surround, bi-scale scheme is used to compute the intensity contrast maps and the edge orientation maps. A great deal of effort is directed towards combining these features into a unique saliency map. In [15], the same authors showed that a simple way to combine the features by normalizing them to a fixed range and summing up, did not work well. In [16], strongly motivated by biological evidence, Itti and Koch use a two dimensional difference-of-gaussians (DoG) to model inhibitory-excitatory interactions between neurons. First, each feature map is re-scaled between 0 and 1. Iteratively, each map is convolved with a DoG filter, the original image is added to the result and the negative entries in the result are set to 0. This “within feature” competitive process produces a conspicuity map (for each feature) with a few peaks left, each peak marking a location in the image most contrasting with its surround. Finally, these three conspicuity maps are summed up linearly and the most salient point in the image is the one with the highest peak.

Another important model of visual attention is the one proposed by Rao, Zelinsky, Hayhoe and Ballard in [34]. The idea here is to define saliency based on iconic representation of each location in the image. An iconic representation at a location is nothing but visual information from a patch in the image, centered at that location. Then, given also an iconic representation of a target object, one could search for that object in the image by defining the saliency at each location as the correlation between the representation at that location and the representation of the model. Encoding

icons literally as raw images, is clearly prohibitively expensive, so the authors propose an efficient way of iconic encoding, as the response around each pixel to a number of different filters, at different scales. Specifically, for a fixed scale, a gaussian is used to generate 3 first order, 2 second order and 4 third order derivatives, for a total of 9 filters. Three scales are considered, yielding a total of 27 multi-scale filters. Each of these filters is applied to each of three (modified) chromatic channels of an RGB image, producing at each location in the image a vector of 81 responses. Given a similar representation of a target object, one essentially defines the saliency at each location (x, y) in the search image as the L^2 norm between the response at (x, y) and the model representation. The next fixation point is selected as the one with the smallest norm. Unlike the Itti and Koch model, which is purely bottom-up and task independent, this model requires knowledge about the task, in that it uses the iconic representation of the target object to define saliency in the search image.

4.2 Reinforcement Learning

Trevor Darrel [9] applies McCallum’s *nearest sequence memory* (NSM) algorithm [27] to implement a gaze control system for gesture recognition. Darrel’s system consists of two cameras: one that provides a static, low-resolution, wide field of view image, and another one with pan-tilt action, which provides high-resolution, narrow field of view images. Part of the world state is fully observable by means of low resolution images. However, the hand gestures and facial expressions can only be observed from the high resolution images provided by the active camera. The various low-level routines can

perform person tracking, and can guide the active camera to saccade from one body part to another, based on the global, low resolution images. Other visual routines perform the gesture recognition based on the high resolution images taken by the active camera, provided that the camera has been directed towards the relevant body parts (e.g. hand or face). The system is supposed to learn how to choose the next fixation point in order to obtain new images of gestures or expressions, and maximally discriminate a particular gesture. The low level routines extract at each time step 9 features from the low resolution and high resolution images: (person-present, left-arm-extended, right-arm-extended, face-expression, left-hand-pose, right-hand-pose, left-hand-foveated, right-hand-foveated, head-foveated). The state is hidden because, for instance, the left-hand-pose (which can be neutral, point or open) cannot be known unless the active camera is pointed to the left hand. At any time, the system can choose between 4 (primitive) actions: look-body, look-head, look-left-hand and look-right-hand. Since the state is hidden, the system can be treated naturally as a *partially observable MDP* (POMDP). The system keeps track of memory sequences $(m_t)_{0 \leq t \leq T}$, where each element m_t is a triple $m_t = (a_t, r_t, o_t)$ of action, reward and observation at moment t , and associates with each such sequence a Q-value, to decide the utility of that sequence in dis-ambiguating the state of the world. The system does manage to learn how to perform the task, but despite the small number of states and actions, it takes a considerable amount of time to train.

An interesting idea is presented by Rimey and Brown [39] where Hidden Markov Models (HMM) are used by an artificial gaze control system to learn saccadic movement sequences. In one example, the pan-tilt camera is assumed to move in small

constant increments in 8 directions. Let v_0, \dots, v_7 be these motions. An 8-state HMM is trained, in which the symbols (observations) are precisely the v_i 's. Observation sequences are now produced, for instance, by drawing a scan path with the mouse on a computer screen, or by monitoring human eye movements. At time $t = 0$ the system “perceives” observation v_{t_0} , at moment $t = 1$ observation v_{t_1} , etc. Thus, a given scan path produces observation sequences which can be used to train the 8 state HMM, i.e. to learn the transition probabilities a_{ij} between states and the observation probability densities $b_j(\cdot)$. The learned HMM is then used to generate symbols from the learned distributions, which drives the camera to moving along a path similar to the one used for training. Obviously this scheme does not use any visual information, so the learned path is independent of the underlying image. However, the authors modify the path generation after the HMM has been trained, to include saccades towards the most salient points in the image. Moreover, they develop the formalism for training *augmented*-HMMS (AHMM). As an application, the authors train two HMMS (a “where” HMM and a “what” HMM), to implement two separate behaviors and combine them using the newly defined notion of AHMM to create a more complex behavior.

Finally, we shall describe a reinforcement learning solution to object recognition due to Bandera et. al. [6]. Suppose we have a foveated vision system whose task is to recognize objects from a finite number of classes. To fix the ideas, suppose the classes are “cube”, “cone” and “pyramid”. Suppose also that the system can detect, fixate and recognize various visual features that are present in all these classes. For instance, a feature could be a 90 degree angle as it is indicative of a cube; vertical

and horizontal edges are also features present in a cube; a circular curve is indicative of a cone, etc. In general, recognition with sufficient confidence requires the presence of a conjunction of these features. The various features have different importance for the classification task, since the more objects share a common feature, the less discriminant power that feature has. The system must learn which features are the most discriminant for the given set of objects. At discrete time steps, the system chooses one feature to interrogate. As the various regions of the object are being fixated on, a confidence measure is associated with each of the features in the “feature database” which indicates the presence or the absence of that feature in the object. The state of the recognition process is defined *intrinsically*, as the vector of these measures. These confidence measures are readily transformed into probabilities and the system must reach a state of minimal entropy by choosing the feature to be fixated next, based on its discriminant power. The one-step reward for choosing a feature is equal to the decrease in entropy it produces when interrogated. A neural net is then used to implement the residual Q-learning algorithm. Although implemented in simulation, we consider this work of great importance, because it is an example of a system in which the states are internal, reflecting the stage of the recognition process, rather than properties of the environment. Moreover, we believe this approach has great potential in automatic feature selection.

Chapter 5

Model Description

5.1 The System Overview

The system consists of a Sony EVI-D30 pan-tilt-zoom camera placed in an elevated location in our lab, controlled through the serial port by a PC with a Pentium II 400 MHz processor. To grab images we use an inexpensive BT848 board. The pan and tilt of the camera (measured in *ticks*) range in the interval $[-860, 862] \times [-281, 283]$. In degrees, these ranges translate into $[-100^\circ, 100^\circ] \times [-25^\circ, 25^\circ]$. At the lowest resolution, the largest field of view of the camera is approximately $48^\circ \times 33^\circ$.

Figure 5.1 shows the architecture of the system. The target object is provided to the system in the form of two template images: one at low resolution, used for coarse search, and the other at high resolution, used for fine matching (see Figure 5.2). The camera takes low resolution images from the lab and feeds them, one at the time, into the two layers of the system. Each low resolution image is processed along two streams - the two main modules of our model (see Figure 5.1). The top

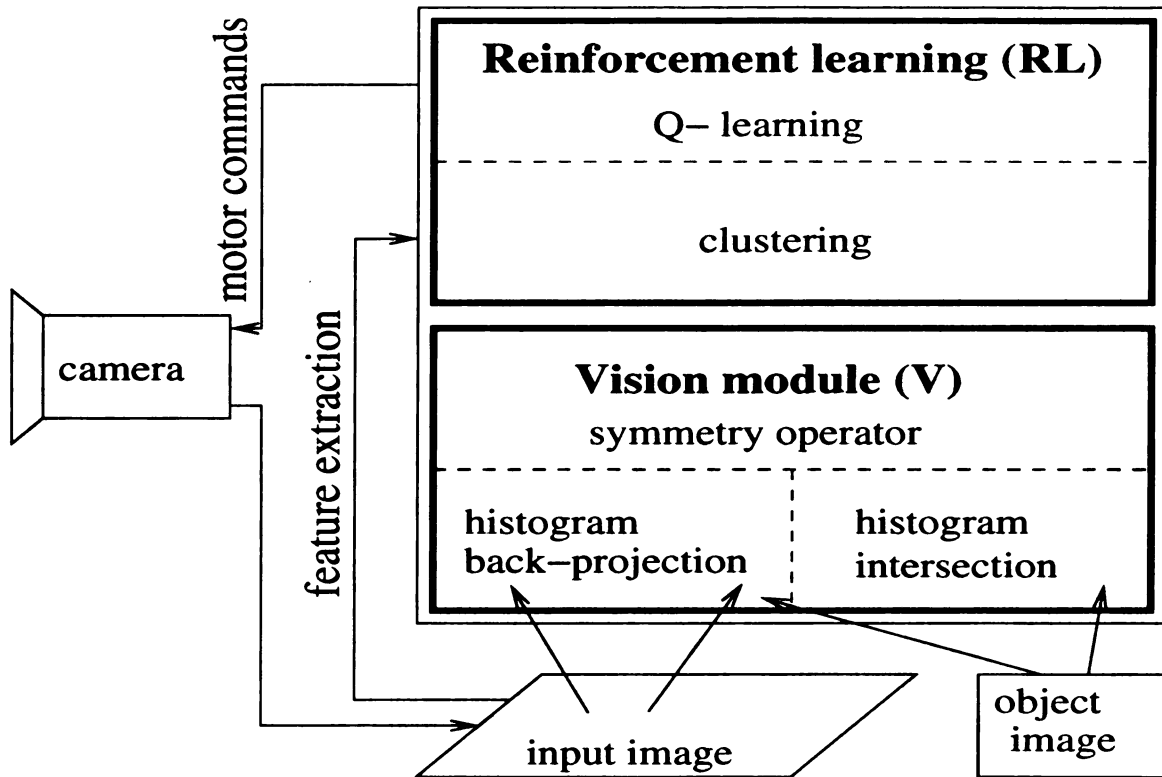


Figure 5.1: Overall architecture of the system. The top module performs the feature extraction necessary to cluster the images online. The resulting regions are used as states in a Q-learning program and a policy for moving towards the target region is learned. The bottom module implements the “within fixation” visual processing, consisting of computation of interesting regions in the image and object recognition.

module (which uses reinforcement learning) learns a set of clusters online consisting of images with similar color histograms. The clusters represent physical regions in the environment, and are used as states in the Q-learning method. This module learns a policy for saccading from one region to another towards the region(s) most likely to contain the target object. By selecting the gaze direction according to its utility (the Q-value), the reinforcement learning module provides top-down control to a lower, purely vision-based module. The vision module consists of low-level visual routines and its purpose is to compute two feature maps (color map and symmetry map) for

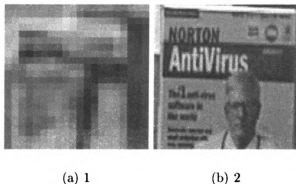


Figure 5.2: (a) Low resolution template of the target object. (b) High resolution template.

representing saliency ¹ in an area of interest in the image, and to recognize the target object, both at low resolution and, if need be, at high resolution for high confidence. All computations in this module are performed in exocentric coordinates ². Unlike [16], at this point we do not attempt to combine the feature maps to form a unique, task independent saliency map, but rather use them sequentially, first the color map (for finding candidate target locations), then the symmetry map (if the target was not found and a new saccade is necessary).

The output is a set of landmarks $L = \{L_1, L_2, \dots, L_n\}$ specific to a particular environment, and a policy on the set L which directs the camera one fixation at the time, starting from any point in the room, and moving towards the region(s) where the target object is normally found. Each element in the set L is a cluster of images representing the same region in the room. The set of landmarks is learned online, and the clustering is solely based on visual input from the camera (i.e. not based on the pan-tilt coordinates of the camera). Although desirable, we don't expect/require

¹The next fixation point is the resulting most salient point.

²Obviously, image coordinates must be transformed to camera coordinates, in order to physically carry out the saccade, but the decision making does not use pan-tilt information.

that the landmarks learned be a faithful representation of the environment (i.e. they may not correspond to physical objects in the room).

5.2 The Q-learning Module

5.2.1 Feature Extraction

In any image classification problem, treating the raw images as vectors is unfeasible due to the very high dimension of the resulting vectors. Dimensionality reduction techniques (e.g. *principal component analysis* [28]) and feature extraction must be applied.

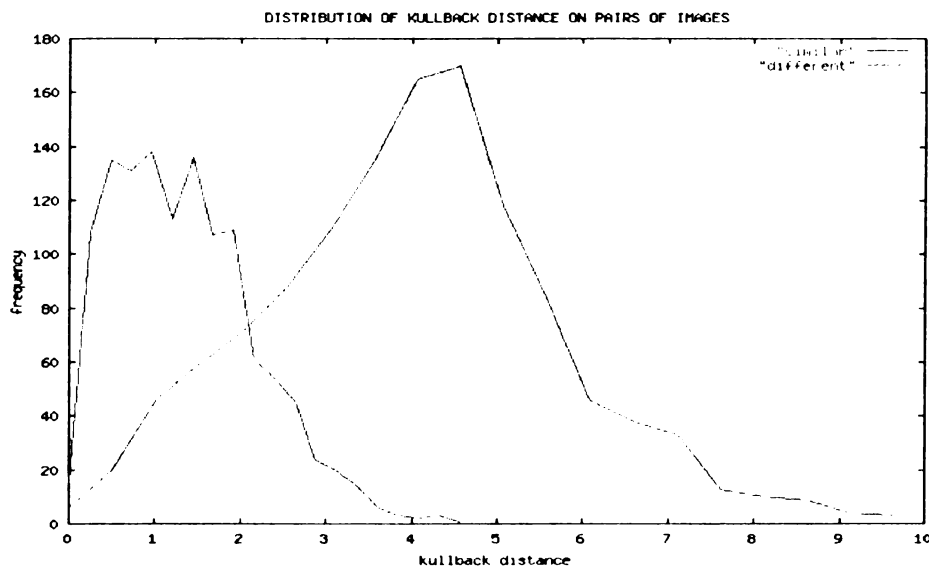


Figure 5.3: Kullback distance of pairs of similar images (red) and on different images (green)

Our first attempt to reduce the size of the observation vectors was to compute the wavelet transform [42] of the images. Wavelet coefficients produce good dimensionality reduction (an order of magnitude better than JPEG compression, with similar

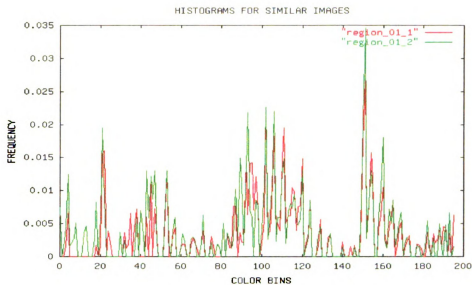


Figure 5.4: Top: feature vectors (4 concatenated color histograms) for two similar images (bottom)

quality), but are too sensitive to small changes in the pan-tilt of the camera, as the wavelet transform is not translation invariant. Small changes in the x,y position of the camera produces large variation in the feature space, which makes any clustering attempt useless.

A set of features that changes slowly as a function of the direction of gaze is provided by the (color) histogram. For an introduction to the computer representation of colors and color spaces see for instance [18], [19], [41]. Binarizing each channel in the RGB space into 16 bins, we get 4096 colors. However, for a given environment,

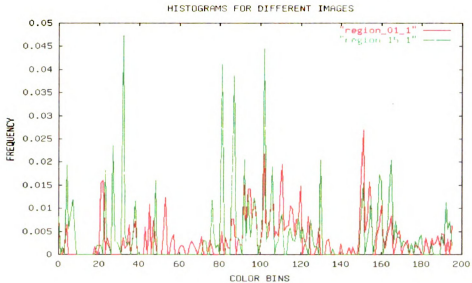


Figure 5.5: Top: feature vectors extracted from two different images (bottom)

many of the colors occur with zero or very small probability. We can then build a histogram on all 4096 colors of 100 random images from the whole lab and retain only the colors with a count of at least 0.5%. The number of colors retained this way was 49. Henceforth, these colors will be called *dominant colors*.

Histograms do have the advantage that they change little when the viewing angle changes, but they may introduce perceptual aliasing: there may be very different images (locations in the environment) which produce the same histogram. To alleviate (but not necessarily eliminate) this problem, we divide each RGB image in 4 quadrants

and concatenate the histograms on the dominant colors on each of the squares. The resulting observation vector has then dimension $196 = 4 \cdot 49$.

Having defined the feature vectors, a suitable metric needs to be chosen, in order to tell whether or not two images are similar. Given a pair (I_1, I_2) of images, two situations can occur: either I_1 and I_2 represent the same region in the environment, or not. If we define two classes

$$C_{similar} = \{(I_1, I_2) | I_1 \simeq I_2\}$$

$$C_{different} = \{(I_1, I_2) | I_1 \not\simeq I_2\}$$

then a metric on pairs of images becomes a feature which can distinguish the two classes. We, therefore, choose a metric that maximizes the inter-class separation. After trying the Euclidean metric and correlation, we chose the symmetric Kullback distance, which we describe next.

The amount of information produced by an event A which occurs with probability $P(A)$ is defined as

$$\log_2 \frac{1}{P(A)} \tag{5.1}$$

Given two probability distribution functions $p(x)$ and $q(x)$ of a random variable x , then the Kullback contrast [17] between p and q is defined as the expected value with respect to q of the difference of information between p and q . That is

$$k(p, q) = E_q(\log_2 \frac{1}{p(x)} - \log_2 \frac{1}{q(x)}) \quad (5.2)$$

$$= \int q(x) \cdot \log_2 \frac{q(x)}{p(x)} dx \quad (5.3)$$

It can be proven [17] that $k(p, q)$ is positive definite, but it is not symmetric. To transform the above Kullback contrast into a metric, we symmetrize it in the standard way:

$$K(p, q) = \frac{k(p, q) + k(q, p)}{2} \quad (5.4)$$

A histogram represents the probability distribution that a certain color occurs in an image, and so, it makes sense to use the Kullback distance to measure the similarity between the two histograms. To see the effectiveness of the Kullback metric, we grabbed 50 random images from our lab by choosing the fixation points uniformly, and 50 images from one region in the lab, i.e. by choosing the fixation points from a gaussian distribution with a small variance, centered around a pre-specified point. We formed all the pairs within each class of images, and for each pair we computed the Kullback distance. Figure 5.3 shows the Kullback distance as a feature on each of the similar/non-similar classes of pairs of images.

The separation threshold between pairs of similar images and non similar images is around 2. However, there is a significant overlap between the two distributions, so,

in order to reduce the false accept rate we moved the threshold to the left, at 1.5, at the expense of the false reject rate.

Figure (5.4) shows a pair of similar images, and their feature vector representation as the concatenation of color histograms on each of the 4 quadrants in each of the images, while Figure (5.5) shows the feature vectors for two non-similar images.

5.2.2 States, Actions and Reward

We define a *state* to be a cluster of images representing the same region in the room. As explained in the previous section, from each image, an *observation vector* is extracted by concatenating the histograms on the dominant colors in each of the 4 quadrants. The similarity metric between two observation vectors is given by the Kullback metric.

Q-learning requires that the set of states be known *a priori*. However, due to the sequential processing of a scene, understanding must be incremental. For this reason, we start with an empty set of clusters of images, and form new clusters as the agent fixates on new regions in the room. The clusters are spherical (with respect to the Kullback distance) , and have a radius of at most 1.5 (see feature extraction). The center of each cluster is the very first observation vector used to initiate creation of a that cluster, and it is not updated as new observation vectors are added.

We define 9 *actions*. A_0 is a saccade to the most salient point in the whole image. Further, we define 8 more actions A_1, \dots, A_8 as follows. Choose a cartesian coordinate system with the origin at the center of the image and for each $k = 1 \dots 8$ let R_k be

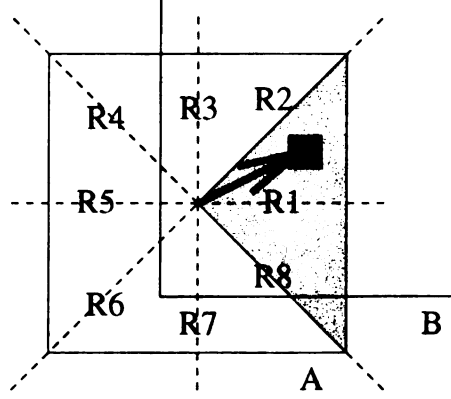


Figure 5.6: The shaded region represents a sector in image A. The most salient point is computed (the blue square) and a new image B, centered at the new location is taken.

the ray through the origin which makes an angle $45^\circ \times (k - 1)$ with the x axis. Let S_k be a 90° sector in the image, bisected by R_k . Then we define action A_k to be the saccade to the most salient point in sector S_k . The shaded region in Figure 5.6 represents sector S_1 in image A. The small blue square is the most salient point in S_1 . Once its coordinates are computed, the camera is directed toward the new point and a new image B is taken. The actions $A_1 \dots A_8$ always force the camera to be directed away from the current fixation point. Action A_0 is justified by biological evidence that human subjects sometimes make a series of very short saccades within the same region of interest, perhaps for the purpose of acquiring more information. If the agent chooses to execute A_0 and if the most salient point is near the optical axis, the agent need not move towards a point in the periphery.

Early in the development of this system we had two extra actions: a search action and a saccade to a random point. We realized, however, that the search for the target object is part of the “within fixation” processing, and consequently, it should be carried out at each time step, and should not be considered a separate action.

The random saccade was also eliminated, because if the outcome of an action has a uniform distribution, then the agent has no way of learning whether it should execute that action or not. Also, before defining the actions using 90° sectors, we tried 45° sectors. However, we have opted in favor of the current size of 90° because by using too narrow angles, some objects might not be entirely comprised within those angles, so their symmetry will be influenced in a negative way.

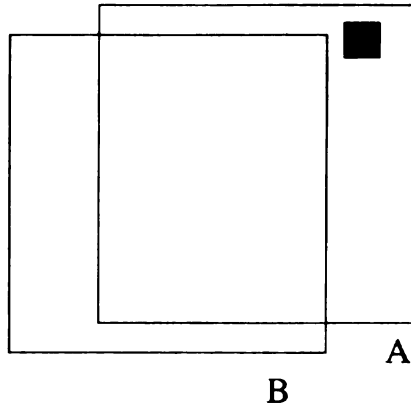


Figure 5.7: If the reward is found in the periphery (image A) then upon subsequent visits to the same region (image B) the reward could be missed, despite the fact that the agent executed the “right” action.

The *reward* function depends on the current state s , the action a taken in s and the next state s' , and is given by:

$$r(s, a, s') = \begin{cases} 100 \cdot e^{-\frac{x^2}{2\sigma^2}} & \text{if target is found in state } s' \\ -4 & \text{otherwise} \end{cases} \quad (5.5)$$

In the above equation, x is the distance from the center of the image (the current fixation point) to the target object and the variance σ controls the size of the gaussian defining the reward. The reason we reward more for bringing the target in the center of the image than for finding it in the periphery, is that the latter situation is unreliable.

Consider the following scenario: the agent is in state s , takes action a and lands in state s' , where it perceives the target somewhere near the periphery. If the agent gets a significant reward, then the action a becomes optimal in state s and will be chosen upon subsequent visits to s . By using solely visual information, the fixation points can only be computed approximately. Thus, often, when the agent revisits state s and takes action a , the target object is just out of the field of view, and the agent gets penalized for the same action that previously brought a large reward. See Figure 5.7.

The *optimality metric* (i.e. the function that the agent tries to maximize) is the expected discounted sum of rewards:

$$V(s) = E\left(\sum_{t \geq 0} \gamma^t \cdot r(s_t, a_{t+1})\right) \quad (5.6)$$

5.3 The Vision Module

The vision module (see Figure 5.1) implements some routines, as part of the “within fixation” processing. We build a color map using *histogram back-projection* [44], [45] to obtain candidate locations for the target object in low resolution/wide field of view images. To assess the value of each of these locations we use a recognizer based on *histogram intersection* [45]. If any of these locations turns out to be a good candidate, the camera is directed to that location, zooms in, and, for high confidence, another match is performed using the same recognizer and the high resolution template of the target object. Finally, if the object was not found in the current frame then a new saccade is necessary. The camera returns to the pan and tilt prior to inquiring

the candidate locations, and a new fixation point is selected. To this end, we use a context free *symmetry operator* [36] to build a symmetry map and select the next fixation as the point with the highest symmetry. For the sake of completeness, we shall briefly describe these routines below.

5.3.1 The Symmetry Operator

Man-made objects tend to have more symmetry than natural objects. In an indoor environment such as a lab, most of the objects are man made, so it is natural to try to employ a symmetry operator in an attempt to fixate on the centers of objects.

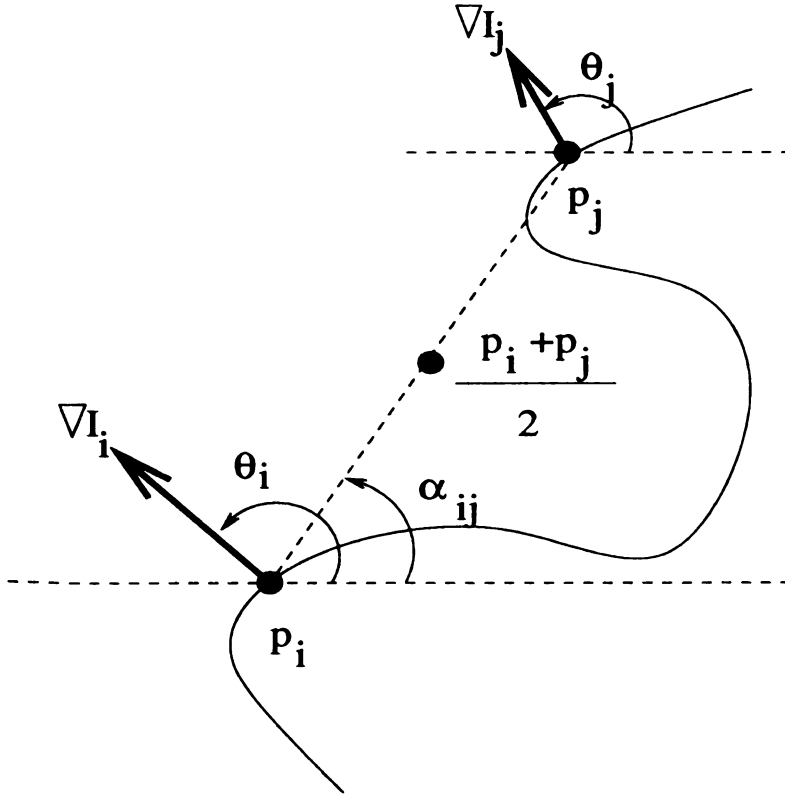


Figure 5.8: Two edge pixels p_i and p_j will vote for their midpoint. The strength of the vote depends on the distance between p_i and p_j , and on the length and the orientation of the intensity gradient at each of the two pixels.

Following [36], for a graylevel image, we compute the edge map, and then we

let each pair of edge pixels vote for their midpoint. The vote is modulated by the distance between the pixels, and by the direction and the length of the gradient of the intensity. More specifically, if $I(x, y)$ is the graylevel image, for each edge pixel p_i define

$$r_i = \log(1 + |\nabla I(p_i)|) \quad (5.7)$$

For any edge pixel p , let θ be the argument of ∇I , i.e. the oriented angle between the horizontal and the vector ∇I . For any two edge pixels p_i and p_j let α_{ij} be the angle between the horizontal and the line $p_i p_j$ (see Figure 5.8).

The phase factor for pixels p_i and p_j is defined by

$$P(i, j) = (1 - \cos(\theta_i + \theta_j - 2 \cdot \alpha_{ij})) \cdot (1 - \cos(\theta_i - \theta_j)) \quad (5.8)$$

We define the distance factor for edge pixels p_i and p_j by

$$D(i, j) = \frac{1}{\sqrt{2\pi}\sigma} \cdot e^{-\frac{(\|p_i - p_j\| - \mu)^2}{2\sigma^2}} \quad (5.9)$$

The vote of the edge pixels p_i and p_j is then defined as

$$V(i, j) = D(i, j) \cdot P(i, j) \cdot r_i \cdot r_j \quad (5.10)$$

The gaussian in equation (5.9) achieves its maximum when the distance between p_i and p_j is equal to a user specified parameter μ , giving pixels at distance μ an increased vote. See Figure 5.9.

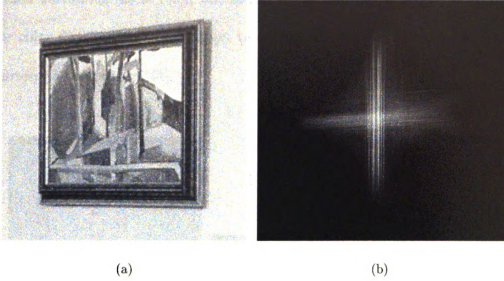


Figure 5.9: (a) Sample image. (b) Symmetry map obtained by maximizing the vote for pixels at distance $\mu = 130$.

By equation (5.8), the phase factor $P(i, j)$ achieves its maximum when

$$\theta_i + \theta_j - 2\alpha = \pi$$

$$\theta_i - \theta_j = \pi$$

which is the case when the two gradients are parallel to the line $p_i p_j$ and point in opposite directions. On the other hand, $P(i, j)$ achieves the minimum when, for instance, $\theta_i = \theta_j$, i.e. when the two gradients are parallel and point in the same direction.

5.3.2 Histogram Back-projection

In addition to symmetry, we use color to define saliency. We do not try to build a task independent, bottom-up color map which would pick up regions contrasting in color

with their surrounds (as in [16]), but rather try to find in the image locations with similar color distribution as the target template. To this end, we use the *histogram back-projection* algorithm [44], [45].

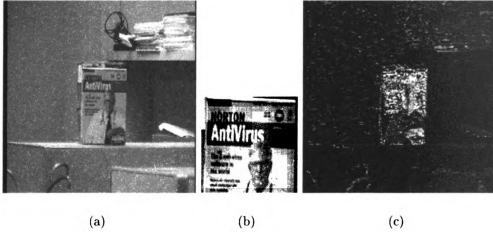


Figure 5.10: (a) Sample search image. (b) Model (target) image. (c) Histogram back-projection of (b) onto (a).

Given two color images I (the search image) and M (the model, provided *a priori*), this algorithm finds the locations in I where M is likely to be found. First, one must build the color histograms H_M and H_I of M and I respectively, on the same set of color bins. For improved performance, we use the L^*, u^*, v^* color space [18] rather than RGB , given by:

$$\begin{aligned}
 L^* &= 25 \cdot \left(\frac{100Y}{Y_0} \right)^{\frac{1}{3}} - 16 \\
 u^* &= 13 \cdot L^* (u' - u_0) \\
 v^* &= 13 \cdot L^* (v' - v_0)
 \end{aligned} \tag{5.11}$$

In the above equations

$$\begin{aligned} u' &= u = \frac{4X}{X + 15Y + 3Z} \\ v' &= 1.5v = \frac{9Y}{X + 15Y + 3Z} \end{aligned}$$

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 0.490 & 0.310 & 0.200 \\ 0.177 & 0.813 & 0.011 \\ 0.000 & 0.010 & 0.990 \end{pmatrix} \cdot \begin{pmatrix} R \\ G \\ B \end{pmatrix} \quad (5.12)$$

and Y_0, u_0, v_0 are the reference white values for Y, u, v , respectively.

The color bins are precisely the colors that occur in I . Two different source images I and I' have different sets of color bins, and so these bins must be re-computed for each pair (I, M) . For an efficient implementation we use look-up tables for the $RGB \rightarrow L^*u^*v^*$ transformation. After computing H_M and H_I we compute the *ratio histogram* R , by

$$R(j) = \min(H_M(j)/H_I(j), 1) \quad (5.13)$$

for each color bin j . This ratio histogram is back-projected onto a gray-level image B (the color map) of the same size as the source image I by

$$B(x, y) = R(I(x, y)) \quad (5.14)$$

That is, for each pixel (x, y) , we determine what bin $I(x, y)$ falls in, we look up the

value of R on that bin, and set $B(x, y)$ equal to this value. The most likely locations of the target object represented by the model M are the pixels (x, y) for which $B(x, y)$ is the largest.

The histogram back-projection algorithm works well if the target object is present in the source image, but it cannot distinguish whether the target is at all present or not. The reason is that in the back-projected image B there is always a maximum. Therefore, after having found the candidate locations in the source image, we must run a recognizer to match the model template M at each location. The recognizer we used is based on the *histogram intersection*.

5.3.3 Histogram Intersection

For the recognition of the target object with sufficient confidence, we use a classifier based on the *histogram intersection* [41] [44].

Let I and T be two images of the same size, that we want to match. Mathematically, neither image has a preferred significance, but in practice T is a high resolution template of a given object (in our case the object is a software box) and I is an unknown image that we want to classify as similar to T or not. As with the histogram back-projection, we compute the color histograms H_I and H_T . Here, since T is given *a priori*, and large enough³, we choose the color bins to be precisely the colors that occur in T . Define the *histogram intersection* distance between H_I and H_T by

³At high resolution an object appears larger.

$$d(H_I, H_T) = 1 - \frac{\sum \min(H_I(j), H_T(j))}{\sum H_I(j)} \quad (5.15)$$

If I and T have the same size, then obviously

$$\sum H_I(j) = \sum H_T(j) = \text{length} \times \text{width}$$

so d is symmetric. Moreover, note that for each color bin j

$$\min(H_I(j), H_T(j)) \leq \sum H_I(j) \quad (5.16)$$

with equality iff $H_I(j) = H_T(j)$. Summing up after j , we get

$$\frac{\sum \min(H_I(j), H_T(j))}{\sum H_I(j)} \leq 1$$

with equality iff all inequalities in (5.16) are equalities, i.e. when $H_I = H_T$. Hence

$$0 \leq d(H_I, H_T) \leq 1 \text{ and } d(H_I, H_T) = 0 \text{ iff } H_I = H_T.$$

The template image T is chosen to be about the size of the target object, and if its histogram is rich enough, a recognizer based on the above distance has very high accuracy. Certainly, given a template T we can produce an image I with the exact same histogram as T by re-shuffling the pixels in T , but this is rarely the case in realistic situations.

5.4 More Visual Routines

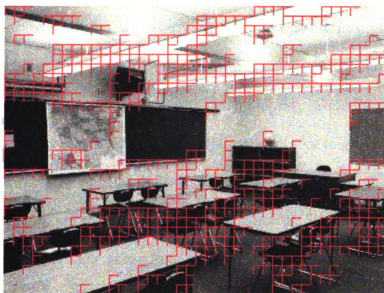
In addition to the feature maps produced using the symmetry operator and the histogram back-projection described in the previous section, we have experimented with a few other feature maps, in an attempt to produce fixation points similar to recorded human fixations. These maps have not been incorporated in the current implementation. Nevertheless, we consider them good starting points in selecting visually important regions in an image, and for this reason we shall describe them briefly in this section.

Variance map. Recordings of human scan patterns [14] show that people almost never fixate on flat, uniform surfaces. Obviously, then, we can use the variance of the intensity to detect non-uniform surfaces: the larger the variance in a region, the more non-uniform that region is. We cover a gray scale image with a grid consisting of square cells. On each cell, we compute the variance of the intensity and we retain only the cells with variance within some interval (v_{min}, v_{max}) . Figure 5.11 shows an indoor image with the recorded human fixations marked by the green dots, and the results of our variance map on the gray scale equivalent of the original image. The red squares are the ones selected by our algorithm. The drawback with the variance map is that the thresholds v_{min} and v_{max} , as well as the size of the cells in the grid must be set a priori by the programmer.

Another idea in defining saliency is to declare a pixel salient if and only if it is sufficiently different from the mean, in some feature (e.g. intensity). To fix the ideas, let $V(x, y)$ be the intensity at pixel (x, y) . Let m_V and σ_V be the mean and the



(a)



(b)

Figure 5.11: (a) Indoor image. The dots represent the recorded human fixation points. Courtesy of the Michigan State University EyeLab. (b) Intensity image of (a). The red squares are the cells with large variance.

standard deviation of the intensity V over the whole image. Construct an **intensity map** I by defining

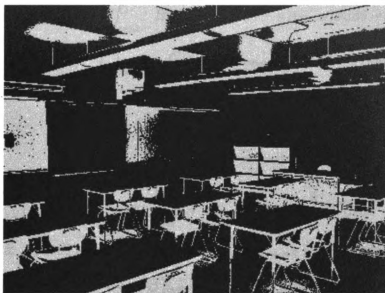
$$I(x, y) = \begin{cases} |V(x, y) - m_V| & \text{if } |V(x, y) - m_V| > \sigma_V \\ 0 & \text{otherwise} \end{cases} \quad (5.17)$$

The intensity map obtained from Figure 5.11 (a) is presented in Figure 5.12 (a). The darker a pixel, the more salient it is. If instead of intensity we use contrast, which we define at each pixel by

$$C(x, y) = \frac{|V(x, y) - m_V|}{m_V}$$

we obtain the map depicted in Figure 5.12 (b), which we term **contrast map**.

The last feature map that we describe is a **color opponency** map. Unlike the previous maps, this map is biologically motivated by neuro-physiological evidence observed in humans, primates and cats [23],[31], [40],[51]. Some of the retinal photoreceptors are highly sensitive to red light (R cones) others to green light (G cones) and yet a third category of cones is sensitive to blue light (B cones). Impulses from these photoreceptors are transmitted to the bipolar cells, which project to the ganglion cells of the retina. The axons from the ganglion cells converge at the optic disk to form the optic nerve, which projects to the *lateral geniculate nucleus* (LGN). It has been observed that some of the ganglion cells (loosely called the red-green opponent cells) have receptive fields consisting of a center of R cones and a surround of G cones. Given an RGB image, we model the red-green opponency by computing at each pixel (x, y) a response



(a)



(b)

Figure 5.12: (a) Intensity map for the image in Figure 5.11 (a). (b) Contrast map. The brighter regions are the most salient. Compare with the human fixations in Figure 5.11 (a).

$$\begin{aligned}
RG(x, y) = & \int \int_{center} R(u, v) g_{center}(x - u, y - v) dudv - \\
& \int \int_{surround} G(u, v) g_{surround}(x - u, y - v) dudv \quad (5.18)
\end{aligned}$$

where $R(x, y)$ and $G(x, y)$ are the red and green values at pixel (x, y) , g_{center} is a gaussian with a small standard deviation and $g_{surround}$ is a gaussian with a larger standard deviation. The first integral in equation 5.18 is the sum of the red values near the pixel (x, y) , weighted by a gaussian peaked at (x, y) and represents the excitatory component. The inhibitory component comes from the surround, and is the sum of the green values weighted by a gaussian of a larger size. See Figure 5.13. We also compute a green-red response GR by using the same procedure. Clearly then, we can iterate this process by letting $R \leftarrow RG$ and $G \leftarrow GR$. The results are presented in Figure 5.14. The brightest spots represent the regions in the input image where the *contrast* between the red center and the green surround (or vice versa) is maximal.

Another color opponency observed in the ganglion cells is the blue-yellow opponency. The receptive fields of these cells are not separated in the center/surround dichotomy as in the red-green cells, but for the purposes of a computer implementation one could use similar techniques.

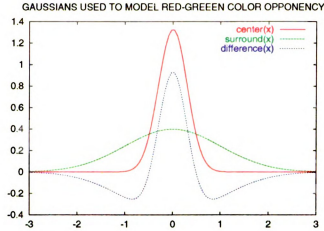


Figure 5.13: Gaussians used to model color opponency. The red curve weighs the responses of the central red cones, while the green curve weighs the (inhibitory) responses of the surrounding green cones. The blue curve is the difference of the two gaussians.

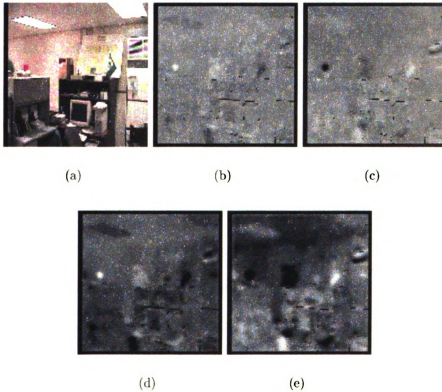


Figure 5.14: (a) Original RGB image. (b) Red-green opponency map. (c) Green-red opponency map. (d) Red-green opponency using (b) and (c) as the red and green channels. (e) Green-red opponency using (b) and (c) as the red and green channels. The brightest regions are the ones in which the center- surround opponency is the largest. The green-red opponency picks up the flag and the computer chair, while the red-green opponency picks up the red box.

5.5 Algorithm

The system starts with an empty set of clusters of RGB images, and with a low resolution template M_{low} and a high resolution template M_{high} of the target object. The camera is pointed at a random initial location and a 384×384 image I_0 is taken at the widest field of view of the camera (hence, at the lowest resolution). An observation vector O_0 is extracted from I_0 (as described in Section 5.2.1) and cluster $C_0 = \{O_0\}$ is formed. An array $Q(C_0, \cdot)$ of length 9 is defined and initialized to 0, and a scalar $T_0 = \infty$.

Suppose now, that after n steps, the system has clusters C_0, \dots, C_m representing various regions in the room, and Q-values $Q(C_0, \cdot), \dots, Q(C_m, \cdot)$. Suppose also that the system has inferred it is in state $C_{current}$ based on the latest low resolution image I_n grabbed by the camera.

An action $a_{current}$ is chosen $\epsilon - greedy$. That is, with probability greater than $1 - \epsilon$, we choose $a_{current} = \arg \max_a Q(C_{current}, a)$, and with probability less than ϵ , a random, exploratory action is chosen.

Execution of action $a_{current}$ is only necessary if the target object is not present in the current image I_n , so the agent tries to detect the object first. To this end, we build a color map B_n by back-projecting the low resolution template M_{low} onto I_n . (see Figure 5.10). *Note:* we do not back-project M_{low} on all of I_n , but on a central patch in I_n , of size 192 (half the size of I_n). The reason is twofold: in the first place, we reduce the processing time; secondly, since in the retina the cone density decreases tremendously towards the periphery [32] [35], color perception is severely

impaired far from the optical axis. Let p_1, p_2, p_3 be the brightest points in B_n . We match the model M_{low} with a neighborhood of p_i for each $i = 1, 2, 3$ using histogram intersection, producing distances d_1, d_2, d_3 . At low resolution, however, histogram intersection is not reliable enough, and the separation threshold varies from frame to frame. To solve this problem, the camera must zoom in at each of the p_i s, grab a new image I'_n and do a fine match, this time at high resolution, using the model M_{high} . However, since histogram back-projection produces candidate locations whether or not the target is present, the camera will have to zoom in three times per frame, most of the time unsuccessfully. We reduce substantially the number of times we zoom in by scoring each of the points p_i by

$$s_i = const \cdot \frac{d_i^2}{B_n(p_i)} \quad (5.19)$$

where $B_n(p_i)$ is the gray level of p_i in B_n (the color saliency), and d_i is the histogram intersection distance (5.15). The lower the score s_i , the better the candidate p_i . Using these scores (rather than just the histogram distances d_i), it is possible to separate the good candidates from the bad ones in most images from the *same* cluster, but we could not find a unique separation threshold *across* clusters. The solution is to have each cluster keep track of its own threshold (call it T) and dynamically adjust it as follows. First, in a newly created cluster set $T = \infty$. At each time step, one of the following two cases can occur: either (a) some of the candidates p_i have scores $s_i < T$, or (b) all $s_i \geq T$. In case (a) the camera is pointed to p_i and zooms in. A high resolution image I'_n is taken and a high resolution model M_{high} is back-projected onto

I'_n . A fine match of M_{high} is performed, again using the histogram intersection. At high resolution, histogram intersection works remarkably well, since images in which the target is present can be readily separated from images which do not contain the target, as can be seen in Figure 5.15

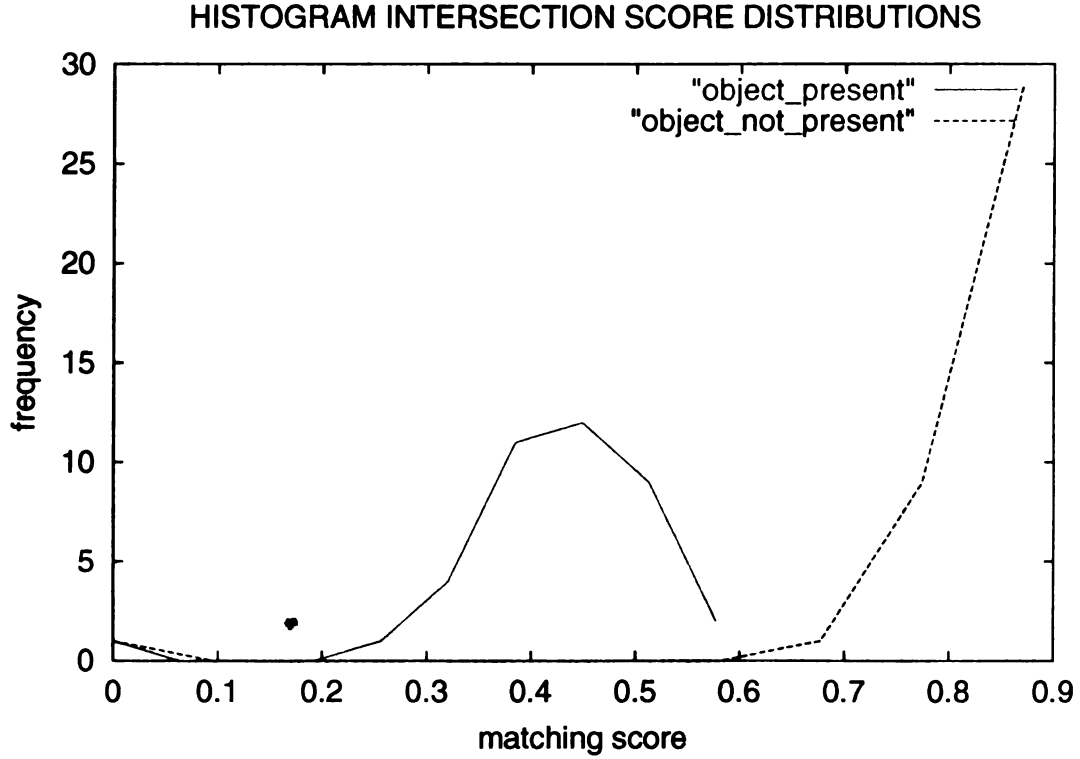


Figure 5.15: Red: distribution of high resolution images containing the target object. Green: distribution of high resolution images not containing the target object.

If the object is found at p_i , the agent is rewarded according to Equation (5.5), and a new training epoch starts with the clusters and the Q-values learned so far, by directing the camera to a new random initial location. Otherwise, if the object is not present at p_i , the camera zooms back out, and proceeds to interrogate the other points p_j with score $s_j < T$, if any. When all candidates p_i have been examined, the threshold T is set to $T = \min\{s_i\}$, the camera returns to the pan and tilt it had before investigating the p_i s and we are in case (b). See Figure 5.16.

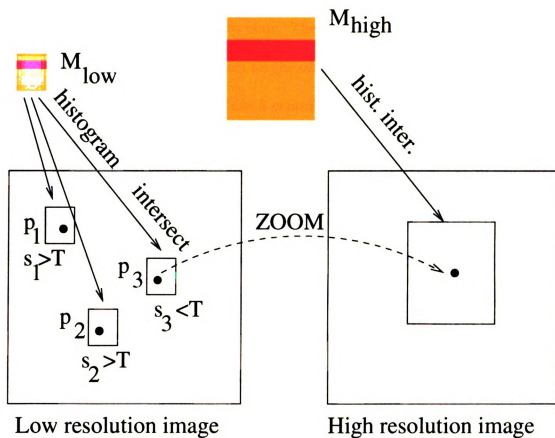


Figure 5.16: Left: Low resolution image. The points p_1, p_2, p_3 are produced using the histogram back-projection of the low resolution model M_{low} . At each p_i , a score s_i is computed using histogram intersection. Right: High resolution image centered at p_3 . The high resolution model M_{high} is first back-projected onto the image, and then matched using histogram intersection. If the target was not found at any of the p_i s then the camera returns to the pan and tilt coordinates it had before the local search, and the threshold T is set to $\min s_i$.

So far, the system has decided that the target object is not present in the current image I_n , either because all candidates p_i had scores $s_i \geq T$, or, because the fine match recognizer ruled out all the candidates. At this point, a new saccade is necessary, so finally, action $a_{current}$ is carried out (from state $C_{current}$). The next fixation point is selected based on the symmetry map. The winner is the point with the largest symmetry vote (Equation 5.10) from all the edge pixels in the image I_n if $a_{next} = A_0$, or, from the edge pixels in one of the 8 central 90° sectors, if $a_{current} = A_k$ for some $k = 1, \dots, 8$. The camera is pointed to the new fixation point and the action is complete.

Knowing now whether $C_{current}$ contains the target or not, the agent is rewarded according to Equation (5.5) for the transition from state $C_{previous}$ to $C_{current}$, as a result of some action $a_{previous}$. Action $a_{current}$ will be rewarded at the next iteration, because only then will the agent know whether $a_{current}$ brought the goal in the field of view or not. $Q(C_{previous}, a_{previous})$ is updated by Equation (2.11). A common technique for speeding up the propagation of the reward is to update more than a single (state, action) pair. Our agent keeps track of the previous five pairs visited, and updates in one step all the corresponding Q-values.

After the update, the camera grabs a new (low resolution) image I_{n+1} , and the agent estimates the state. More specifically, a feature vector O_{n+1} is extracted from I_{n+1} , and the closest cluster to O_{n+1} is found (with respect to the Kullback distance (5.4)). Denote this cluster by $C_{closest}$, and its center by $O_{closest}$. If the Kullback distance $K(O_{n+1}, O_{closest}) < k_{min} = 1.5$, then O_{n+1} is assigned to cluster $C_{closest}$, and the agent is in state $C_{next} = C_{closest}$. No adjustments are made to $C_{closest}$, so

that at any time, each cluster consists of a single vector, namely the initial vector used to create that cluster. If $K(O_{n+1}, O_{closest}) > k_{max} = 2.5$, then the current observation is not sufficiently close to any of the existing clusters, and a new cluster $C_{next} = \{O_{n+1}\}$ is created. With this new cluster, a scalar $T_{next} = \infty$ is defined. Uncertainty occurs when $k_{min} \leq K(O_{n+1}, O_{closest}) \leq k_{max}$, since this is the case when the overlap between the two curves in Figure 5.3 is significant. In a situation like this, it is not clear whether the images that produced the observations O_{n+1} and $O_{closest}$ should be considered similar, or non-similar. The system could very well be “looking” at a previously visited region, but is unable to recognize it, due to noise. Certainly, we could simply create a new cluster whenever $K(O_{n+1}, O_{closest}) \geq k_{min}$, but in doing so, the system unnecessarily ends up keeping multiple clusters representing the same region in the environment. Therefore, before creating a new cluster, the system gets a second chance at estimating the state, by making a small number of very short saccades around the current fixation point. If still none of the new observations falls within distance k_{min} from the closest cluster, then the system has no choice but create a new cluster. A new action is chosen and the process repeats until either the object is found, or until a maximum number of iterations has been reached.

In what follows, we use the term “epoch” to refer to a sequence of at most $n_{MAX} = 100$ fixations. If either the goal has been reached, or all 100 time steps have elapsed, a new epoch is started by pointing the camera to a random location, in order to ensure sufficient exploration of the whole state space. We summarize below the algorithm used for a single training epoch. Some of the technical details are omitted in order to keep the presentation simple.

1) Initialization. Choose the gaze direction of the camera randomly. Set the number of iterations $n = 0$. Get a low-resolution image I and extract an observation vector O (see Section 5.2.1). Define a cluster $C_0 = \{O\}$, define $T_0 = \infty$, and an array $Q(C_0, \cdot) = 0$ of Q-values.

WHILE object not found and $n < n_{MAX}$

2) Choose an action $a_{current}$, $\epsilon - greedy$.

3) **Color map:** back-project a low resolution target template onto I . Score each of the top 3 candidates p_1, p_2, p_3 by (5.19). Zoom in at each p_i with $s_i < T_n$, grab a high resolution image, and match the high resolution target template using the histogram intersection. If object found, go to step 6. Else, zoom out, adjust $T_n = \min\{s_i\}$ and go to next step.

4) **Symmetry map:** compute symmetries on the sector corresponding to action $a_{current}$ and direct the camera towards the most salient point in that sector.

5) **Clustering:** get new feature vector O . Find the closest cluster $C_{closest}$ to O in Kullback distance K . If K is sufficiently small, $C_{next} = C_{closest}$. Else, if K is sufficiently large, create a new cluster $C_{next} = \{O\}$, define $T_{next} = \infty$ and initialize an array $Q(C_{next}, \cdot) = 0$. Finally, if K is near the decision boundary (see Figure 5.3), make a few short saccades and re-compute K . If the new image still cannot be unambiguously classified, create a new cluster.

6) **Q-update:** compute reward for action $a_{previous}$ using equation (5.5). Update the last 5 Q-values according to equation (2.11). Set $n \leftarrow n + 1$.

END WHILE

Chapter 6

Experimental Results

We describe in this section a number of experiments designed to determine in the first place if any visual learning is at all possible with our reinforcement learning agent, and if so, how good the learning was. We also compare the performance of our agent with the performance of a random agent.

6.1 Learning a Policy for Finding the Target

In the first experiment, we trained the agent to learn in which direction to direct its gaze in order to reach the region where the target object is most likely to be found, in trials of 400 epochs each. Every 5-th epoch was used for testing, i.e. the agent simply executed the policy learned so far. The performance metric was the number of fixations to the goal. Within a single trial the starting position was the same in all test epochs.

Figures 6.1 and 6.2 show the number of steps to goal for two initial starting

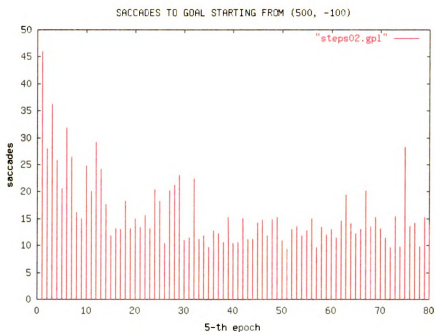


Figure 6.1: Average number of fixations for every 5-th epoch over five trials of 400 epochs each. At every 5-th epoch, the policy learned was tested starting from pan=500, tilt=-100.

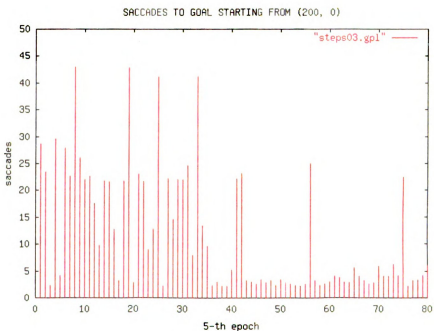


Figure 6.2: Average number of fixations for every 5-th epoch over five trials of 400 epochs each. At every 5-th epoch, the policy learned was tested starting from pan=200, tilt=0.

positions. The results were averaged over several trials. It is apparent that in general the number of fixations decreases with the number of epochs. Occasionally, there are long fixation sequences towards the end of the trial, but the agent recovers quickly.

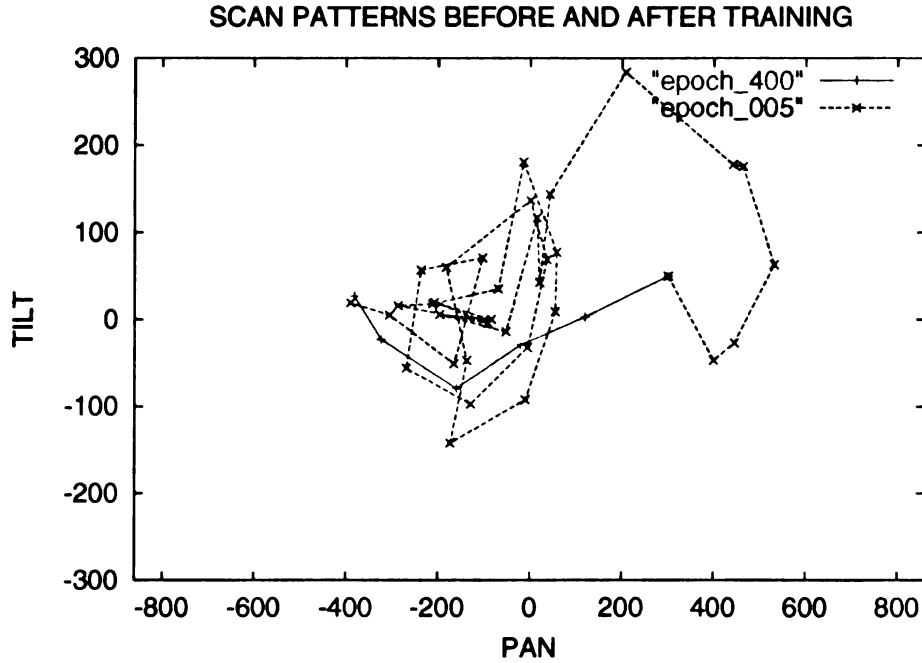


Figure 6.3: Initial scan path (green) and learned scan path (red). The rectangle $[-860, 860] \times [-290, 290]$ represents the actual pan-tilt range of the camera. The search starts at (300, 50) and the target object is found around $(-380, 30)$.

We plotted in Figure 6.3 two sample scan paths, one at the beginning of learning, which is very convoluted and has a large number of fixations, and another one after the system was trained to find the object. Figure 6.4 shows a sequence of regions “as seen” by the camera, as it fixated from the starting position (rightmost image) to the target (leftmost).

Figure 6.5 plots the centers of the learned regions (states) (the blue dots) in the pan-tilt coordinate system as well as the best actions in each region. Only states with more than 15 visits are shown, out of a total of 104 states. The position of

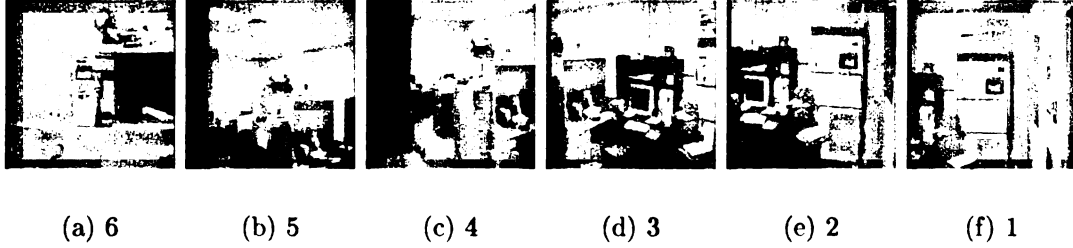


Figure 6.4: The sequence of fixations corresponding to the learned path (red) in Figure 6.3. The camera starts from region (1) and gradually moves towards the goal region (5), Here, a sufficiently good candidate is found at low resolution (1), the camera zooms in (6) and does a high resolution match. This sequence of images corresponds to the red scan path in Figure 6.3

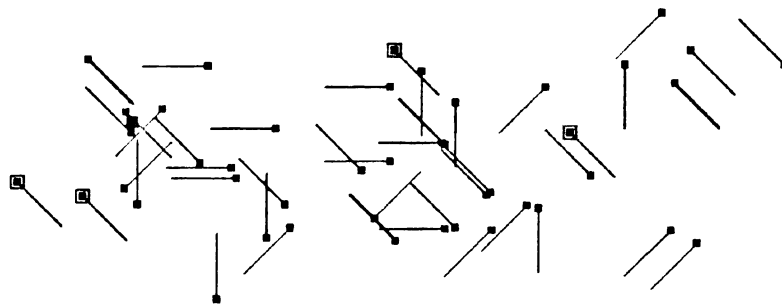


Figure 6.5: States learned (the blue squares) and the best actions. Action A_0 is represented by a red square around a state. Actions $A_1 - A_8$ are represented by a red segment originating at the state. The target is the green square.

the target is marked by the green square. Action A_0 is represented as a red square around the state and actions $A_1 - A_8$ are represented as a red segment originating from the center of the state. Note that the length of each segment does not represent the length of the saccade. The actual saccade was directed at some salient point in a sector around a red segment. This explains why the field of the red segments does not point directly and uniformly towards the target (the green square).

6.2 Executing the Learned Policy

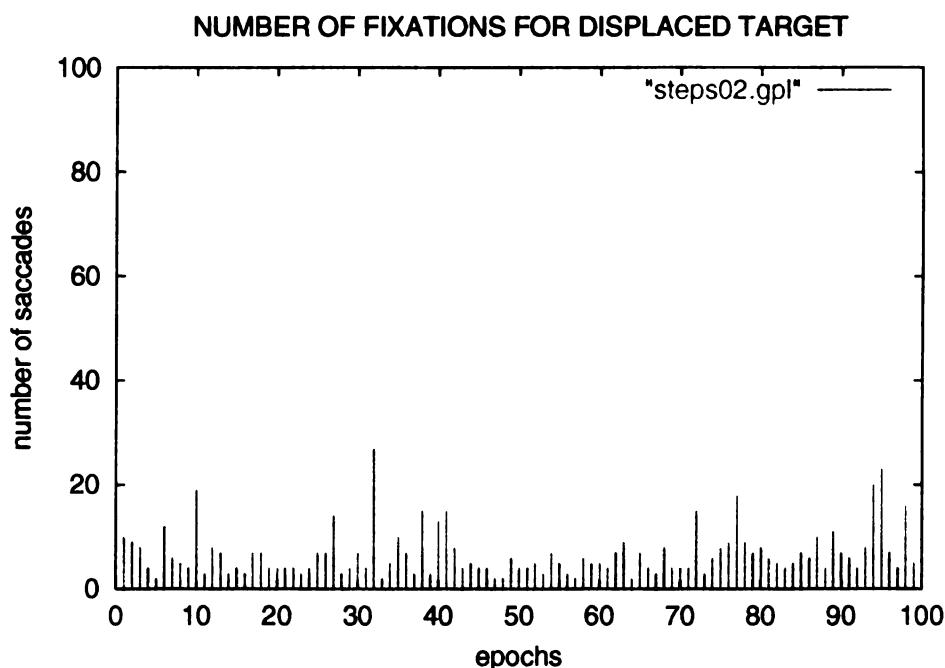


Figure 6.6: Number of saccades per epoch in finding the object inverted, displaced by 60 cm from the training position. Testing from $\text{pan}=500$, $\text{tilt}=-100$.

The purpose of our second experiment was to determine what changes in the location of the target the system could manage. After the agent was trained to find the target upright in its usual location, the object was turned upside down and

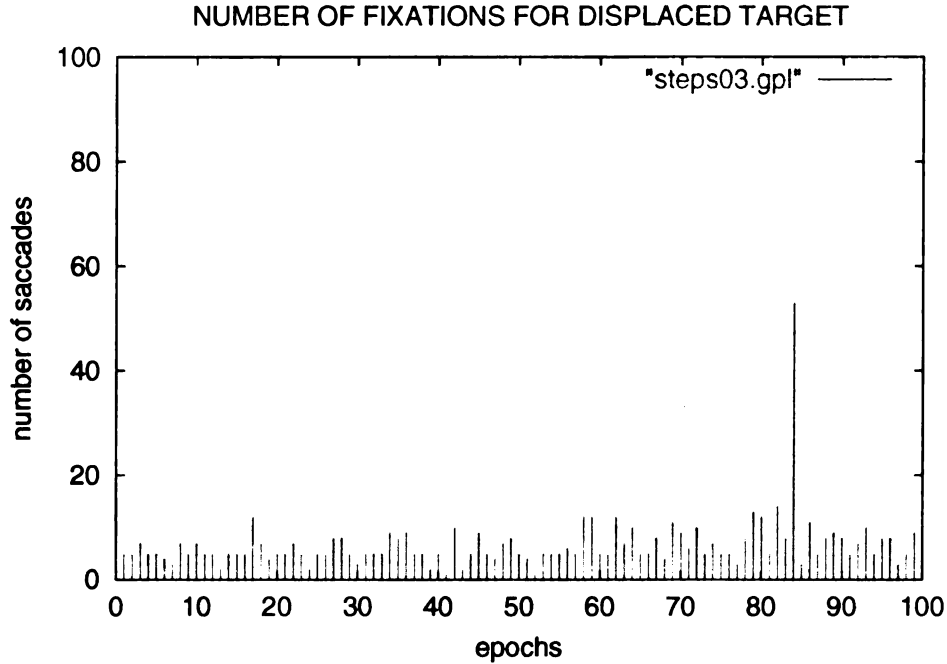


Figure 6.7: Number of saccades per epoch in finding the object inverted, displaced by 60cm from the training position. Testing from pan=200, tilt=0.

displaced to a new location (about 50-60 cm from the training position), while the agent was executing the learned policy. Again, we tested starting from a few initial locations and we averaged over several test trials. The agent successfully found the object in a small number of fixations, as shown in Figures 6.6 and 6.7.

6.3 Re-training for a Different Target

To estimate how sensitive the system was to the various parameters in the visual routines, we re-trained the agent to find a different object (the green flag in Figure 6.8) *without changing any parameters*. Somewhat surprisingly, there is no noticeable qualitative difference between the performance in this experiment and in experiment 1. Figures 6.9 and 6.10 show the number of fixations for finding the flag, starting from

$(500, -100)$ and $(200, 0)$, respectively.



Figure 6.8: (a) The target object in the third experiment was the green flag. (b) Low resolution template. (c) High resolution template.

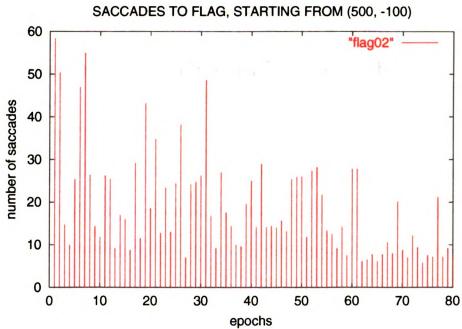


Figure 6.9: Average number of fixations for finding the MSU flag (Figure 6.8) for every 5-th epoch over five trials of 400 epochs each. At every 5-th epoch, the policy learned was tested starting from $\text{pan}=500$, $\text{tilt}=-100$.

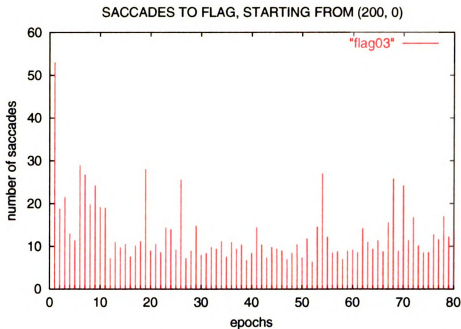


Figure 6.10: Average number of fixations for finding the MSU flag (Figure 6.8) for every 5-th epoch over five trials of 400 epochs each. At every 5-th epoch, the policy learned was tested starting from pan=200, tilt=0.

6.4 Comparison with Random Search

Finally, in our fourth experiment, we compared the performance of the reinforcement learning agent, with a random agent i.e. an agent which performed random search. The actions in the random agent were exactly the same as in the reinforcement learning agent, but their selection was random, rather than based on the Q-values. The results are presented in Figures 6.11 and 6.12. On average, the number of fixations in the random search was between 50 and 70, substantially larger than the average number of fixations in the previous experiment (6-7 fixations), where the agent was only executing a learned policy.

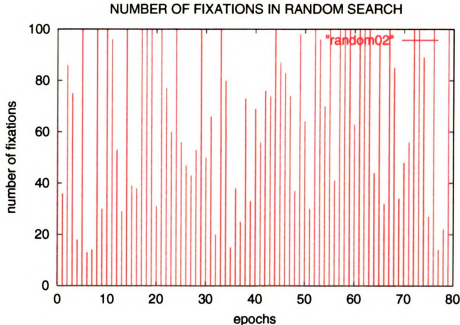


Figure 6.11: Number of saccades per epoch in a random search. Testing from pan=500, tilt=-100.

6.5 Comparison with Exhaustive Search

Besides the comparison with the random agent, we shall present, for the sake of completeness, a comparison between the learning strategy proposed in this thesis, with an exhaustive search agent. By exhaustive search we mean that the camera starts exploring its pan-tilt universe from one fixed location (e.g. the upper left corner) and moves in fixed increments one step at the time, until the whole pan-tilt space has been searched, or until the target has been found. To this end, let us recall that the pan-tilt range of our camera is $[-100, 100] \times [-25, 25]$ degrees, and that the widest field of view for one frame is about 48×33 degrees. We assume that all the visual processing is the same in the search agent as in the learning agent, so first a low resolution search is carried out, and then a high resolution search, around the most

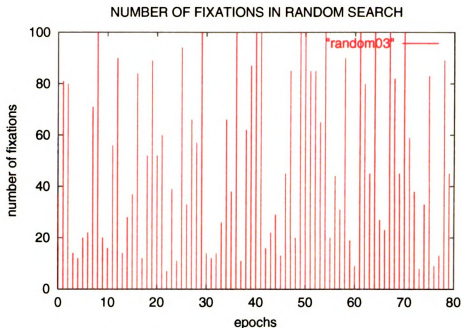


Figure 6.12: Number of saccades per epoch in a random search. Testing from pan=200, tilt=0.

promising locations detected at low resolution, as described in Section 5.5. Recall from the same section, that within one frame, the search is not performed on the whole image, but rather on a central patch of half the size. Thus, each low resolution color search covers a visual field of about 24×17 degrees. The pan-tilt range of the camera can then be tiled by approximately $8 \times 3 = 24$ search regions. As the target object could be in any of these 24 regions, the system would make at most 24 fixations for finding the object, in the worst case, and $(1 + 2 + \dots + 24)/24 = 12.5$ fixations on average. These results are summarized in Table 6.1.

Start	Training (box)	Test (box)	Training (flag)	Random	Exhaustive
(500, -100)	15.8	6.7	18.7	66.9	worst: 24
(200, 0)	11.41	6.6	12.8	51.1	avg: 12.5

Table 6.1: Average number of fixations for the three experiments starting from two test points. In experiment 1 testing was done every 5-th epoch. The rest of the epochs were training epochs. Experiment 2 consisted only in executing a learned policy, with the target object inverted and slightly displaced. In experiment 3, a random search was done.

Chapter 7

Discussion

The set of experiments presented in the previous section is by no means comprehensive. No systematic analysis is done to explore the effect of the learning parameters (the learning rate α , and the exploration rate ϵ), nor to observe the effect of the various parameters from the visual routines. Although no further tuning was necessary to find the flag, after the software box was found, we do not expect this to be the case for any target object. What the experiments do show, however, is that learning does occur, and that the system is able to find the target reliably, and that the search is robust to affine transformations (translations and rotations) of the object within one region. We discuss below some technical issues that we encountered.

Visual saliency. Color and symmetry were the means of defining saliency in our implementation. The symmetry operator used picks up objects fairly well, provided the rough size of the objects is known. Tuning the operator for objects of a particular size, will not work well for objects of different sizes. This is one reason why the fixations of our agent, in general, do not cluster on objects, the way human fixations

do. Moreover, the symmetry map is influenced by the fact that we apply the symmetry operator on a rather limited sector of the image. Symmetric objects that simply do not fit in that sector, will have lower saliency.

Although back-projection histogram is a good way of detecting objects by color, it is not biologically plausible. As the evidence for color opponency shows, biological vision systems use *feature contrast*, rather than *feature values*. Histograms, of course, use values. Some effort was required to distinguish genuine candidate locations of the target object from noise, in low resolution images, but back-projection histogram in conjunction with the histogram intersection is a fast and effective way of building a color saliency map, provided a template of the target object is given.

Clustering. To our knowledge, an artificial general purpose classifier with nearly perfect rate does not exist, and we did not attempt to build one. Due to the probabilistic nature of Q-learning, however, an imperfect (but reasonable) classifier will suffice. If the classifier has accuracy of, say, 90%, then, statistically, the agent will receive the correct reward *most* of the time, which is all it needs to learn the correct best action.

We opted to represent images by means of color histograms, in order to categorize pairs of images as “similar” or “non-similar”. The loss of the spatial distribution of the colors in an image leads to perceptual aliasing, but we found that perceptual aliasing is unlikely to occur in a cluttered environment such as a lab, particularly if we subdivide the image into smaller pieces, and compute the histogram on each piece. The real problem that we believe is at the core of our clustering algorithm, is more mathematical in nature. Let V be the feature space produced by our feature

extraction method. Each observation is a vector in V and has dimension 196 in this implementation. For each pan-tilt (x, y) of the camera, we get a vector $O(x, y) \in V$, namely the observation extracted from the image taken when the camera is pointed at (x, y) . For a given environment, we get this way a mapping from the two dimensional euclidean space into V

$$O : \mathbf{R}^2 \rightarrow V$$

hence a surface in V . Since the histogram is known to change slowly with (x, y) , this surface is continuous, which entails that there are no natural clusters in V to represent different regions in the room. By choosing a threshold for the similarity metric (the Kullback distance), we effectively used spherical clusters in the feature space V .

We preferred to lower the Kullback threshold used to cluster the low resolution images in order to reduce the false accept rate (i.e. to reduce the number of images classified as “similar” when in fact they were “non-similar”). Inherently, this increased the false reject rate, that is, often two genuinely similar images were classified as non-similar, and a new cluster was started, representing in fact a region that had been visited previously. After 400 training epochs, we ended up with about 80-100 clusters, out of which about 40 were distinct by visual inspection. Although “duplicate” clusters are wasteful and unnecessarily increase the number of states, they are harmless for the Q-learning algorithm, *unless* their number increases indefinitely, because in that case, the (state, action) pairs would not be visited sufficiently many

times.

The way we propose to approach this problem is to invoke an alternative similarity metric when (and only when) the Kullback distance is indecisive. Certainly there will be an interval (a, b) , in which the alternative metric cannot distinguish well between similar and non-similar image, just like the Kullback metric does not work well between $(1.5, 2.5)$, but it is crucial that (a, b) and $(1.5, 2.5)$ be disjoint. Good candidates would be metrics on edge pixels (edge density, curvature, etc.), because edge information and color information are uncorrelated.

An alternative idea¹ is to modify the feature extraction. Instead of dividing each image into quadrants and compute the color histogram on each of the 4 regions, we can extract visual information (e.g. color histogram, or some iconic representation in the sense of [4]) locally, from around the top salient points. This way, the visual information will be more “anchored” to the environment, and consequently, less sensitive to (small) changes in the camera pan and tilt. The problem that this approach raises is that, unlike in the case of the 4 quadrants where we can always impose a fixed order, there is no canonical way of defining an ordering on salient points, which is absolutely crucial if local observation vectors are going to be concatenated to produce an observation vector for the whole image.

¹Jon Connell, IBM, personal communication.

Chapter 8

Conclusions and Future Work

We have built a reinforcement agent which learns the region where a given object can usually be found. Learning occurs by choosing the direction of the next saccade according to its utility (the Q-value). Once the direction has been chosen by the reinforcement learning module (see figure 5.1), low level visual routines find the most salient point in that direction and aim the camera at it.

The experimental results show that the agent is capable of learning, even with an unsophisticated reinforcement learning mechanism. In comparison with random search, our agent is about ten times better (see table 6.1), which shows the usefulness of learning. Certainly, exhaustive search is always an option, but it defeats the purpose of our study.

Our agent comes short of learning spatial relationships between objects. It cannot, for instance, learn that a painting is usually on a wall, or that a pen is usually on a desk. It can learn, however, relationships of the form *this* pen is in *this* region (which happens to be a table).

In the current implementation, the information that is being integrated across fixations is color information (through histograms), and the direction of the next saccade. No deep image understanding takes place at each fixation, except to decide whether the current region has been seen before, and whether or not it contains the target object. In this sense, the information we retain is minimal, but sufficient for the search task.

The visual information used to define saliency in an image is based on color and symmetry. If the symmetry operator is task independent and context free, the back-projection histogram is not, in that it uses a template image of the target object.

The performance of our system can be improved in a number of ways. In the first place, different variants of Q-learning should be investigated. For instance, one could use *eligibility traces* [43], which are temporary records of occurrence of an event (e.g. visiting a (state,action) pair), used to selectively assign credit to the events that were marked as eligible. Secondly, the bottom-up visual attention mechanism must be improved for a better selection of the fixation points. We proposed in Section 5.4 a number of other feature maps, but understanding how to combine them into a unique saliency map requires further investigation. Thirdly, the classifier used to cluster images should be improved by adding one more feature, “orthogonal” to the Kullback metric, or by using iconic representations around the top salient points in the image, as already discussed at the end of Chapter 7. Our clustering algorithm tries to group together similar images, irrespective of their utility. A more sophisticated algorithm [25] which clusters observations using both a metric in feature space and the Q-values could be used for improved performance.

We view this work as an initial exploration towards the implementation of a gaze control mechanism on a mobile robot. We purposefully avoided the use of pan-tilt information in the decision making process, because due to the robot motion, the target object (such as an exit sign, or an elevator door) changes its position with respect to the agent. A more sophisticated recognizer capable of identifying the target at different distances, and from various points of view will be needed.

We foresee some interesting applications of this system on a real robot. For instance, by *visually* finding a fixed object in a room, the robot could automatically align itself, which is a necessary step before navigation. Or, the robot could simply find the door, and exit the room. A more complex behavior would be to train the search system to recognize several objects in several rooms, and then have the robot tell what room it is in, based on the confidence with which it finds the expected objects.

Bibliography

- [1] S. Ahmad and S. Omohundro. Efficient visual search: A connectionist solution. In *Proceedings of the 13th Annual Conference of the Cognitive Science Society*, Chicago, 1991.
- [2] J. Aloimonos, A. Bandopadhyay, and I. Weiss. Active vision. In *Proceedings First International Conference on Computer Vision*, pages 35–54, London, 1987.
- [3] S.M. Anstis. A chart demonstrating variations in acuity with retinal position. *Vision Research*, 14:589–592, 1974.
- [4] D.H. Ballard. Animate vision. *Artificial Intelligence*, 48:57–86, 1991.
- [5] D.H. Ballard, M.M. Hayhoe, and J.B. Pelz. Memory representations in natural tasks. *Journal of Cognitive Neuroscience*, 7(1):66–80, 1995.
- [6] C. Bandera, F. Vico, J. Bravo, M. Harmon, and L. Baird. Residual q-learning applied to visual attention. In *Proceedings of the 13th International Conference on Machine Learning*, pages 20–27, Bari, Italy, July 3rd-6th 1996.
- [7] JR. Bergen and B. Julesz. Focal attention in rapid pattern discrimination. *Nature*, 303:696–698, 1983.

- [8] C. Bushnell, M.E. Goldberg, and D.L. Robinson. Behavioral enhancement of visual response in monkey cerebral cortex. i. modulation in posterior parietal cortex related to selective visual attention. *Journal of Neurophysiology*, 4:775–772, 1981.
- [9] T. Darrel and A. Pentland. Active gesture recognition using partially observable markov decision processes. In *13-th IEEE Intl. Conference on Pattern Recognition*, 1996.
- [10] R. Desimone and J. Duncan. Neural mechanisms of selective visual attention. *Annual Review of Neuroscience*, 18:193–222, 1995.
- [11] E.E. DeYoe and D.C. Van Essen. Concurrent processing streams in monkey visual cortex. *Trends in Neurosciences*, 11(5):219–226, 1988.
- [12] M.E. Goldberg and R.H. Wurtz. Activity of superior colliculus in behaving monkey. *Journal of Neurophysiology*, 35:560–574, 1972.
- [13] P. Haenny, J. Maunsell, and P. Schiller. Cells in prelunate cortex alter response to visual stimuli of different behavioral significance. *Perception*, 13:A7, 1984.
- [14] J.M. Henderson and A. Hollingworth. *Eye Guidance in Reading and Scene Perception*, chapter Eye Movements During Scene Viewing: An Overview, pages 269–295. Elsevier Science B.V., 1998.
- [15] L. Itti and C. Koch. A comparison of feature combination strategies for saliency-based visual attention systems. In *SPIE Human Vision and Electronic Imaging IV (HVEI'99)*, pages 473–482. 1999.

- [16] L. Itti and C. Koch. A saliency-based search mechanism for overt and covert shifts of visual attention. *Vision Research*, 40:1489–1506, 2000.
- [17] M. Jagersand. Saliency maps and attention selection in scale and spatial coordinates. an information theoretic approach. In *Proc of 5-th International Conference on Computer Vision*, pages 195–202, 1995.
- [18] A. K. Jain. *Fundamentals of Digital Image Processing*. Prentice Hall, 1989.
- [19] R. Jain, R. Kasturi, and B.G. Schunck. *Machine Vision*. McGraw-Hill, Inc., 1995.
- [20] B. Julesz and J.R. Bergen. Textons, the fundamental elements in preattentive vision and perception of textures. *Bell Syst Tech Journal*, 62:1619–1645, 1983.
- [21] L.P. Kaelbling. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.
- [22] C. Koch and S. Ullman. Shifts in selective visual attention: Towards the underlying neural circuitry. In *Human Neurobiology*. 1985.
- [23] P. Lennie, P.W. Haake, and D.R. Williams. The design of chromatically opponent receptive fields. In M.S. Landy and J.A. Movshon, editors, *Computational Models of Visual Processing*, pages 71–82. MIT Press, 1991.
- [24] S. Mahadevan. Average reward reinforcement learning: Foundations, algorithms and empirical results. *Machine Learning*, 22:159–195, 1996.

- [25] S. Mahadevan and J. Connell. Automatic programming of behavior-based robots using reinforcement learning. *Artificial Intelligence*, 55(2-3):311–365, June 1992.
- [26] D.C. Marr. *Vision*. Freeman, 1982.
- [27] A.K. McCallum. *Reinforcement Learning with Selective Perception and Hidden State*. PhD thesis, University of Rochester, 1996.
- [28] B. Moghaddam and A. Pentland. Probabilistic visual learning for object representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):696–710, 1997.
- [29] B.A. Olshausen, D.C. Van Essen, and C.H. Anderson. A neurobiological model of visual attention and invariant pattern recognition based on dynamic routing of information. *Journal of Neuroscience*, 13:4700–4719, 1993.
- [30] J.K. O’Regan and A. Levy-Schoen. Integrating visual information from successive fixations: Does trans saccadic fusion exist? *Vision Research*, 23(8):765–768, 1983.
- [31] O. Packer, A.E. Hendrickson, and C.A. Curcio. Photoreceptor topography of the adult pigtail macaque (*macaca nemestrina*) retina. *Journal of Comparative Neurology*, 288:165–183, 1989.
- [32] S.E. Palmer. *Vision Science - Photons to Phenomenology*. MIT Press, 1999.
- [33] M. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley & Sons, 1994.

- [34] R.P.N. Rao, G.J. Zelinsky, M.M. Hayhoe, and D.H. Ballard. Eye movements in visual cognition: A computational study. Technical Report 97.1, University of Rochester, 1997.
- [35] K. Rayner. Eye movements in reading and information processing: 20 years of research. *Psychological Bulletin*, 124(3):372–422, 1998.
- [36] D. Reisfeld, H. Wolfson, and Y. Yeshurun. Context free attentional operators: The generalized symmetry transform. *International Journal of Computer Vision*, 14:119–130, 1995.
- [37] R.A. Rensink. Seeing, sensing and scrutinizing. *Vision Research*, 40:1469–1487, 2000.
- [38] R.A. Rensink, J.K. O'Regan, and Clark J.J. To see or not to see: The need for attention to perceive changes in scenes. *Psychological Science*, 8:368–373, 1997.
- [39] R.D. Rimey and C.M. Brown. Controlling eye movements with hidden markov models. *International Journal of Computer Vision*, 7(1):47–65, 1991.
- [40] R.W. Rodieck. Quantitative analysis of cat retinal ganglion cell response to visual stimuli. *Vision Research*, 5:583–601, 1965.
- [41] L. Shapiro and G. Stockman. *Computer Vision*. Addison-Wesley, 2000.
- [42] G. Strang. Wavelets and dilation equations: A brief introduction. *Siam Review*, 31:613–627, 1989.
- [43] R.S. Sutton and A.G. Barto. *Reinforcement Learning*. MIT Press, 1998.

- [44] M.J. Swain and D.H. Ballard. Object identification using color cues. Technical report, University of Rochester, NY, 1990.
- [45] M.J. Swain and D.H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.
- [46] J.K. Totsos, S.M. Culhane, W.Y.K. Wai, Y. Kai, N. Davis, and F. Nuflo. Modelling visual attention via selective tuning. *Artificial Intelligence*, 78:507–545, 1995.
- [47] A. Triesman. The role of attention in object perception. In *Physical and Biological Processing of Images*. 1983.
- [48] A. Triesman and G. Gelade. A feature-integration theory of attention. *Cognitive Psychology*, 12:97–136, 1980.
- [49] L. Ungerleider and M. Mishkin. Two cortical visual systems. In *Analysis of Visual Behavior*, pages 549–585. 1982.
- [50] C. Watkins. *Learning From Delayed Rewards*. PhD thesis, King’s College, 1989.
- [51] A.B. Watson. Neural contrast sensitivity. In M.S. Landy and J.A. Movshon, editors, *Computational Models of Visual Processing*, pages 95–107. MIT Press, 1991.
- [52] L.E. Wixson. Exploiting world structure to efficiently search for objects. Technical report, University of Rochester, July 1992.

- [53] J. Wolfe, K. Cave, and S. Franze. Guided search: An alternative to the feature integration model for visual search. *Journal of Experimental Psychology: Human Perception and Performance*, 15:419–433, 1989.
- [54] A.L. Yarbus. *Eye Movements and Vision*. Plenum, 1967.

MICHIGAN STATE UNIVERSITY LIBRARIES



3 1293 02094 8182