A Reinforcement Learning Model of Selective Visual Attention

Silviu Minut
Autonomous Agents Lab, EB3210
Department of Computer Science
Michigan State University
East Lansing, MI 48823
minutsil@cse.msu.edu

Sridhar Mahadevan
Autonomous Agents Lab, EB3210
Department of Computer Science
Michigan State University
East Lansing, MI 48823
mahadeva@cse.msu.edu

ABSTRACT

This paper proposes a model of selective attention for visual search tasks, based on a 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 agent 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. Detailed experimental results show that the number of saccades to a target object significantly decreases with the number of training epochs. The results also show the learned policy to find the target object is invariant to small physical displacements as well as object inversion.

1. MOTIVATION

The problem of *visual search* is to find a small object in a large usually cluttered environment (e.g. a pen on a desk). In solving such a problem it is preferable to use wide field-of-view images. 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 [1]. 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 receptors (and with it the visual acuity as well) decreases exponentially from the fovea towards the periphery. To make up 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 $900°/s$), ballistic motions called *saccades*. *Fixations* are the periods between saccades during which the eyes remain relatively fixed, the visual information is processed and the location of the next fixation point is selected. Figure 1 shows human scan patterns in an indoor scene recorded with an eye-tracker. Note how the fixation points (the green points) tend to cluster on objects in the image and avoid large flat uniform surfaces (e.g. walls, ceilings, floors, desktops, etc.).

Figure 1: Recorded human scan patterns. Fixations (the green dots) tend to cluster on objects. Courtesy of John M. Henderson, Eye-Lab, Michigan State University

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 direct the gaze at the most visually salient object.

From an engineering standpoint, a sequential attention mechanism is a very attractive approach, because it has the potential of requiring only sparse local models [2]. However, the visual attention mechanism raises a plethora of

AGENTS'01, May 28-June 1, 2001, Montréal, Quebec, Canada.

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 Ulmann [8] propose a saliency map theory which is a task independent, bottom-up model of visual attention. In this framework, Itti and Koch [6], extract three types of feature maps (a color map, an edge map and an intensity map) and fuse them together in a unique map (termed *saliency* map). However, the selection of the next fixation must require some top-down control since low-level visual information is not usually 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. [14] 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 integrate across saccades, to produce a global understanding of the scene. We propose an overall model that integrates top-down gaze control with bottom-up reactive saliency map processing, based on reinforcement learning.

2. PROBLEM DEFINITION AND GENERAL APPROACH

Given an object and an environment (see Figure 2), we address the problem of how to build a vision agent that learns where that object is most likely to be found, and how to direct its gaze towards the object. The system must produce a set of landmarks $\{L_0, L_1 \dots L_n\}$ (regions in the room) and a policy on this set, which leads the camera towards the most probable region containing the target object.

The idea is to learn the approximate region where the target can normally be found, solely based on visual information (i.e. no pan-tilt information) [1].

Our general approach is to use reinforcement learning to select the coarse direction of the saccades. Although reinforcement learning has been proposed as a promising approach for the problem of gaze control in vision (e.g. [3]), to our knowledge, few, if any, previous systems have been implemented that found actual objects in real cluttered environments. Darrel [4] used reinforcement learning in a task related to gesture recognition. However, this system assumed a set of feature detectors that are task-dependent (e.g. person-present?, smile? etc.) which were computed using both a low-resolution wide field-of-view camera and a high-resolution narrow field-of-view camera. The policy learned by reinforcement learning in Darrel's system is based on these high-level feature detectors. Our approach, by contrast, does not require any high level feature detectors, and the policy learned through reinforcement learning is based on the actual images seen by the camera.

Once the direction has been selected, the precise location of the next fixation point is determined by means of visual saliency. Unless the object is detected in the current field-of-view, a new saccade is carried out. The camera takes low resolution/wide field-of-view images [2] at discrete time intervals. The system tries to recognize the region in the image, as well as whether or not the target object is present, by using a low resolution template of the object. Since reasonable detection of an object of the size of a book or smaller, in a room, is difficult at low resolution, we can only hope to get some *candidate* locations for the target object. We simulate foveated vision by zooming in and grabbing high resolution/narrow field-of-view images centered at the candidate locations, and compare them with a high resolution color template of the target. (Figure 2).

Figure 2: Color template of the target object, and the environment in which it must be found.

Prior to describing the model and its functionality in Section 5, we shall sketch the basic ideas in the reinforcement learning paradigm and how we can exploit them for building a visual search agent. We also briefly explain the visual routines that we use towards defining saliency in an image and for recognition of the target object.

3. REINFORCEMENT LEARNING

We adopt the framework of reinforcement learning in this paper, as a model of sequential decision-making [12]. In this approach, an agent is embedded in an environment, which it perceives to be in one of many possible states. In each state, the agent can choose and execute an action. After each action, the agent re-estimates the state by observing the environment, and gets rewarded for the action it just completed. Given a specific task, the goal is to learn an optimal action in each state, in order to accomplish the task. The agent starts in some state s_0, takes an action a_1 and reaches some state s_1 (possibly equal to s_0), and so on. In general, for any time step t, the transition from s_t to s_{t+1} could depend on the previous history (i.e. on the sequence $s_0, a_1, \dots, s_t, a_{t+1}$), but if it only depends on s_t and a_{t+1}, the system is said to have the *Markov property*.

To estimate the state, the agent gets an observation (a vector) describing the environment. If based on that vector, the agent can uniquely determine the state, and the environment is Markov, then the resulting sequential decision-making problem can be modeled as a *Markov Decision Process* (MDP). An MDP is a system (S, A, P, R), where S is the set of *states* of the system, and A is the set of actions available to the agent. $P = \{P(s'|s,a)\}$ is the set of probabilities which govern the transitions from state s to s' as a result of action a, and $R = \{r(s,a,s')\}$ is the set of rewards. A *(stationary) policy* is a function $\pi : S \longrightarrow A$. The agent follows a policy π if in any state s it always chooses the same action $\pi(a)$.

The goal of the agent is to find a policy which is optimal with respect to some *objective function*. One well-known

[1] Our goal is to extend this approach to a camera mounted on a mobile robot, in which case the target object may have *any* (pan,tilt) coordinates with respect to the camera.

[2] For our camera, the largest field-of-view is approximately 48×33 degrees. The human field-of-view is approximately 180°.

optimality metric is the expected value of the sum of discounted rewards:

$$\begin{aligned} V^{\pi}(s) &= E_{\pi}(\sum_{t=1}^{\infty} \gamma^{t-1} \cdot r(s_{t-1}, \pi(s_{t-1}))) \\ &= r(s, \pi(s)) + \gamma \cdot \sum_{s' \in S} P(s'|s, \pi(s)) V(s') \quad (1) \end{aligned}$$

where $s_0 = s$ and γ (the discount factor) is a constant between 0 and 1.

Optimizing this objective function amounts to finding a policy π^*, such that $V^* \stackrel{def}{==} V^{\pi^*}$ satisfies the optimality equations

$$V^*(s) = \max_{a \in A}(r(s,a) + \gamma \cdot \sum_{s' \in S} P(s'|s,a) V^*(s')) \;\forall s \in S \quad (2)$$

The agent may, or may not know a priori the transition probabilities and the reward. In the first case, the agent has a model of the problem, and dynamic programming techniques (value iteration, policy iteration) could be used to compute an optimal policy [10]. In the visual search problem, the transition probabilities and the reward are not known to the agent. A model-free *Q-learning* algorithm [15] can be used to find optimal policies. Essentially, with each state-action pair (s, a), the agent stores a value $Q(s, a)$. Initially, the matrix Q is initialized to 0 (or randomly). Iteratively, the Q-values are updated by the rule

$$\begin{aligned} Q_{n+1}(s,a) &= (1-\alpha) Q_n(s,a) + \\ & \alpha(r(s,a) + \gamma \cdot \max_{a' \in A} Q_n(s', a')) \quad (3) \end{aligned}$$

where s is the state and a is the action at iteration n and α is a learning rate parameter. Q-learning is proven to converge [15], provided all pairs (s, a) are visited infinitely often. If we denote the limit by Q^*, then optimal policies can be obtained by defining

$$\pi^*(s) = \arg\max_a Q^*(s, a)$$

3.1 States, Actions and Rewards

States Recorded scan patterns from human subjects show that people fixate from object to object [5]. It would, therefore, be natural to define the *states* of an artificial vision agent as objects in the environment. In doing so, we face a paradox: objects must be recognized as worth attending to, before they are fixated on. However, an object cannot be recognized prior to fixation, since it is perceived at a low resolution (e.g. faces in the visual periphery cannot usually be recognized).

We define states as clusters of images representing the same region in the environment. We represent each image essentially by color histogram on a reduced number of bins. These bins are specific to a given environment, and represent only the colors that occur more often than a certain percentage (0.5%). In the cluttered lab environment in which the experiments reported in this paper were carried out, we were left only with 48 colors. Obviously, the use of histograms introduces perceptual aliasing, as two different images could have identical histograms. To reduce the aliasing, rather than representing each image by a single global histogram, we compute the histograms on each of the quadrants, in order to retain some of the spatial distribution of colors. The observation vector has dimension $196 = 4 \times 48$ and is the concatenation of these 4 color histograms. Certainly, this does not eliminate the aliasing, for two completely white, but different walls, would still look the same using this representation. However, natural environments are sufficiently rich, and perceptual aliasing in such environments is often minimal. We use the symmetric Kullback distance as a measure of how different two images are. The Kullback contrast between two probability distributions p and q is defined as

$$\begin{aligned} k(p,q) &= E_q(\log_2 \frac{1}{p(x)} - \log_2 \frac{1}{q(x)}) \\ &= \int q(x) \cdot \log_2 \frac{q(x)}{p(x)} dx \quad (4) \end{aligned}$$

It can be proven [7] 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:

$$K(p,q) = \frac{k(p,q) + k(q,p)}{2} \quad (5)$$

To determine the separation threshold between pairs of similar images and pairs of dis-similar images, we collected a set of similar images, by choosing 50 fixation points from a gaussian distribution with small variance, and a set of dis-similar images, by choosing 50 fixation points uniformly in the whole room. We computed $K(I, J)$ for all pairs, and plotted the resulting distributions for the two classes of pairs (similar/non-similar). The results are shown in Figure (3).

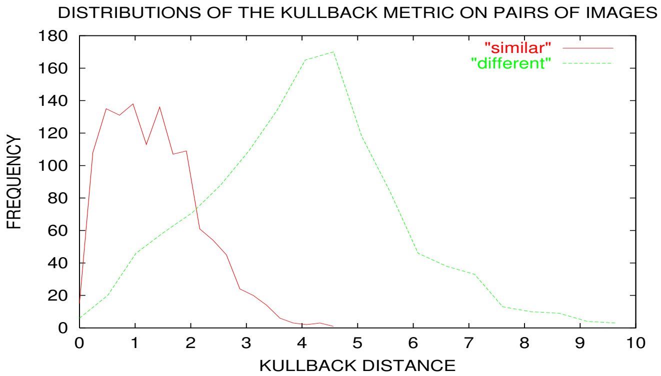


Figure 3: Distribution Kullback distances of pairs of similar images (red) and on different images (green)

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 alarms we moved the threshold to the left, at 1.5.

Actions. We define actions $A_1, \ldots A_8$ as saccades to the most salient point in one of eight 90° equally spaced central angles in the image. We explain in the next section how saliency is computed. In addition to these actions, the agent can also choose action A_0, which is a saccade to the most salient point in the whole image.

Reward. The agent receives positive *reward* for a saccade which brings the target object in the field-of-view (the closer to the optical axis, the higher the reward), and a small negative reward otherwise. Specifically, if x is the distance from the target object to the center of the image and σ is the scale factor, then

$$r(s,a,s') = \begin{cases} 100 \cdot e^{-\frac{x^2}{2\sigma^2}} & if \quad target \ found \ in \ state \ s' \\ -10 & otherwise \end{cases} \tag{6}$$

This function rewards the agent preferentially for finding the object using short saccades (near the target). Longer saccades are given less reward since they tend to be more unreliable in finding the target (as the target is more displaced from the optical axis, and consequently might not be in the frame of reference upon a subsequent visit).

4. WITHIN-FIXATION PROCESSING

The "within fixation" processing comprises computation of two components. The first component is a set of two feature maps implementing low-level visual attention, used to select the next fixation point. The second component is a recognizer, used at low resolution for the detection of candidate target locations, and at high resolution for high confidence recognition of the target.

Histogram intersection [13] can be used to match two images I and M (of the same size) using color information. Given two color histograms h_I and h_M on the same number of bins, we define the similarity measure

$$d(h_I, h_M) = 1 - \frac{\sum \min(h_I(i), h_M(i))}{\sum h_I(i)} \tag{7}$$

It can be shown (see [13]) that d is a distance if $\sum h_I(i) = \sum h_M(i)$ (which is always the case if the two images have the same size). It is difficult to define a threshold based on which to discriminate between pairs of similar images and pairs of dis-similar images, if both I and M vary. For this reason, we did not use the histogram intersection for clustering. However, if one of the images (the model M) is fixed , then a reliable threshold can be found. This is the case when we search for a pre-specified object.

As with the histogram intersection, the **histogram back-projection** was also introduced in [13]. Given two images I (the search image) and M (the model), we want to locate M in I. To this end, we compute the color histograms h_I and h_M on the same number of color bins, and then a ratio histogram

$$R(i) = min(\frac{h_M(i)}{h_I(i)}, 1) \tag{8}$$

The ratio histogram is then back-projected on a blank gray-scale image B of the same size as the search image I. This operation requires one pass through I: for each pixel (x, y) we set $B(x, y) = R(j)$ iff $I(x, y)$ falls in bin j.

Histogram back-projection works very well on realistic images (see Figure 4), but has the disadvantage that it always produces candidate locations, even if the target is not present. The actual presence of the target at the candidate locations must be decided using a recognizer, our choice being histogram intersection.

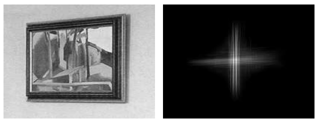
We use a **symmetry operator** in order to fixate on objects, since most (man-made) objects in a room have vertical, horizontal or radial symmetry. The symmetry operator, introduced in [11], computes an edge map first, and then has each pair p_i, p_j of edge pixels vote for its midpoint by

(a) (b) (c)

Figure 4: (a) Sample search image. (b) Model (target) image. (c) Histogram back-projection of (b) onto (a).

$$vote(p_i, p_j) = D(p_i, p_j) \cdot P(p_i, p_j) \cdot r_i \cdot r_j \tag{9}$$

where D is a distance factor (gaussian in terms of $||p_i - p_j||$), P is a phase factor, which is maximal for pixels whose gradients are parallel and point in opposite directions, and $r_i = \log(1 + ||\nabla I||)$. See Figure (4).



(a) (b)

Figure 5: (a) Sample image. (b) Symmetry map.

5. MODEL DESCRIPTION

Each low resolution image is processed along two streams - the two main modules of our model (see Figure 6). The top 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 toward 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 second module consists of low-level visual routines and its purpose is to compute two feature maps (color map and symmetry map) for representing saliency [3] in an area of interest in the image, and to recognize the target object, both at low resolution and at high resolution. All computations in this module are performed in exocentric coordinates [4]. Unlike [6], 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).

[3]The next fixation point is the resulting most salient point.

[4]Obviously, 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.

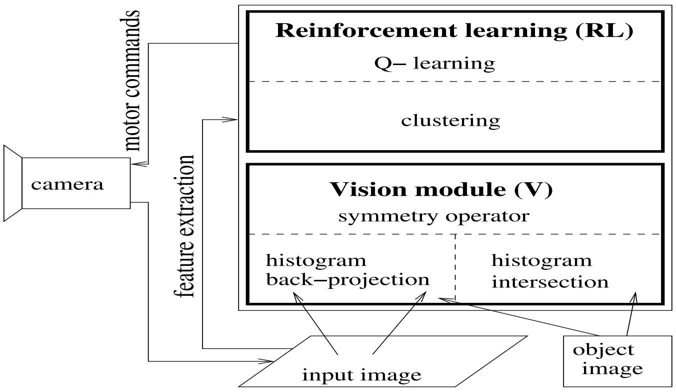


Figure 6: Overall architecture of the visual search agent.

Our color map is the histogram back-projection B of the (low-resolution) target template M on the current (low-resolution) frame (see Figure 4). Let p_1, p_2, p_3 be the brightest points in B. We match the model M 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_is, grab a new image and do a fine match. 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

$$s_i = const \cdot \frac{d_i^2}{g_i} \qquad (10)$$

where g_i is the gray level of p_i in B (the color saliency), and d_i is the histogram intersection distance (7). The lower the score s_i, the better the candidate p_i. Using the score (rather than just the histogram distance 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 $T = \infty$. Since all s_i are less than the threshold, the camera zooms in at each p_i. If the target was not found at any of the p_i, then set $T = \min\{s_i\}$. Upon subsequent visits to that region, locations with scores larger than T need not be viewed at high resolution.

At each time step, if all the points have a score $s_i > T$ (in which case the camera does not zoom in), or if the camera does zoom in, but it does not find the target at any of the points p_is, then a new saccade is necessary. At this point we make use of the symmetry map. Depending on the request from the reinforcement learning module, the next fixation point is selected as the point with the largest symmetry vote (equation 9).

An "epoch" is a sequence of at most 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:

- Initialize 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 (as described in Section 3.1).
- Define a cluster $c_0 = \{O\}$ to contain just I, define $T_0 = \infty$, and an array $Q(c_0\ ,\ \cdot\ )$ of Q-values, initialized to 0.
- **While** object not found and $n < 100$
  - **1)** Compute the color map by back-projecting a low resolution target template onto I. Score each of the top 3 candidates p_1, p_2, p_3 by formula (10).
    - ∗ If some score s_i is less than the threshold T_n
      - · Saccade to p_i, zoom in and match the high resolution target template using histogram intersection. If object found, go to step 5.
  - **2)** If object not found
    - ∗ Adjust the threshold T_n by $T_n = \min\{s_i\}$.
    - ∗ With prob. $> 1 - \epsilon$ choose next action A_i to maximize $Q(c_n, .)$ (else select random action)
    - ∗ Compute the symmetry map on the sector corresponding to action A_i.
    - ∗ Direct the camera toward the most salient point in that sector.
  - **3)** Get a new image I (low-resolution) and extract the feature vector O.
  - **4)** Find the closest cluster $c_{closest}$ to O in Kullback distance.
    - ∗ If the shortest Kullback distance is sufficiently small, classify I as belonging to cluster $c_{closest}$. Otherwise, create a new cluster $c_{new} = \{O\}$, define $T_{new} = \infty$ and initialize an array $Q(c_{new}\ ,\ \cdot\ ) = 0$.
  - **5)** Compute reward for action performed using equation (6)
  - **6)** Update the last 5 Q-values by equation (3).
  - **7)** Set $n \leftarrow n + 1$
- **end while**

6. EXPERIMENTAL RESULTS

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 7 and 8 show the number of steps to goal for two initial starting positions. The results were averaged over

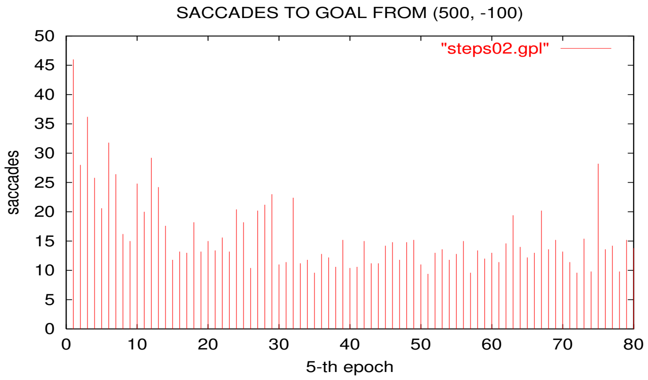


Figure 7: Average number of fixations for every 5-th epoch) over five trials of 400 epochs each. At every 5-th epoch, the policy learned was tested starting from pan=500, tilt=-100.

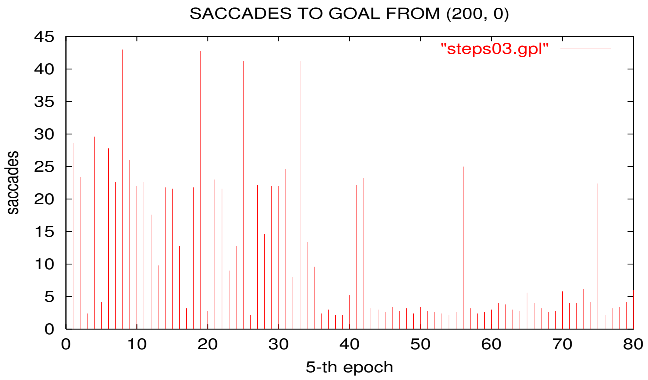


Figure 8: Average number of fixations for every 5-th epoch) over five trials of 400 epochs each. At every 5-th epoch, the policy learned was tested starting from pan=200, tilt=0.

several trials. It is apparent that in general the number of fixations decreases with the number of epochs. Occasionally, there are long fixation sequences toward the end of the trial, but the agent recovers quickly.

We plotted in Figure (9) 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 (10) shows a sequence of regions "as seen" by the camera, as it fixated from the starting position (rightmost image) to the target (leftmost).

The Kullback classifier had a classification rate well over 90%, which produced "clean" clusters (i.e. clusters containing almost exclusively images representing the same region). Often, however, the agent would fixate on a previously visited region, and would not recognize it as visited, starting a new cluster. This behaviour is due obviously to the increased sensitivity of the Kullback threshold, which would pick up even the smallest variations in illumination.

After having trained the agent to find the target object upright, we tested separately if it could find the object upside down, and in several slightly different locations (about 60 cm) from the original training position. Again, we tested

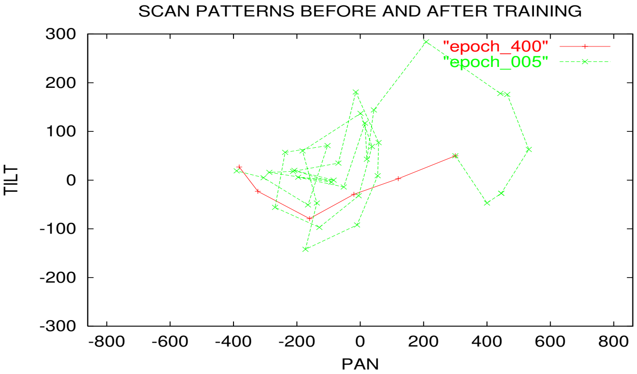
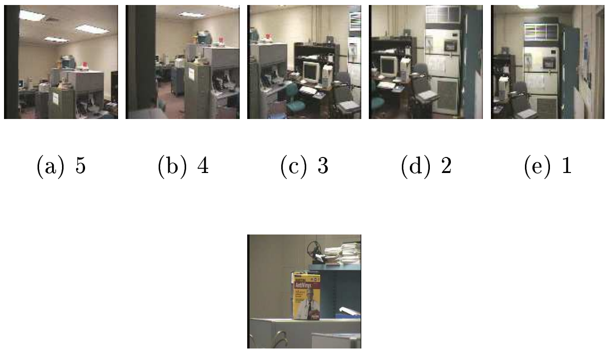


Figure 9: Initial scan path (green) and learned scan path (red). The rectangle $[-860, 860] \times [-290, 290]$ represents the actual pan-tilt range of the camera. The search starts at $(300, 50)$ and the target object is found around $(-380, 30)$.



(a) 5 (b) 4 (c) 3 (d) 2 (e) 1

(f) 6

Figure 10: The sequence of fixations corresponding to the learned path (red) in Figure 9. The camera starts from region (1) and gradually moves toward 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.

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 (11) and (12).

Finally, comparison with random search is in order. We evaluated the performance of the agent with the RL module suppressed (but still fixating on the salient regions produced by the vision module). The results over 400 epochs are presented in Figure 13. The agent managed to find the object after 50-60 saccades (on average), depending on the starting position.

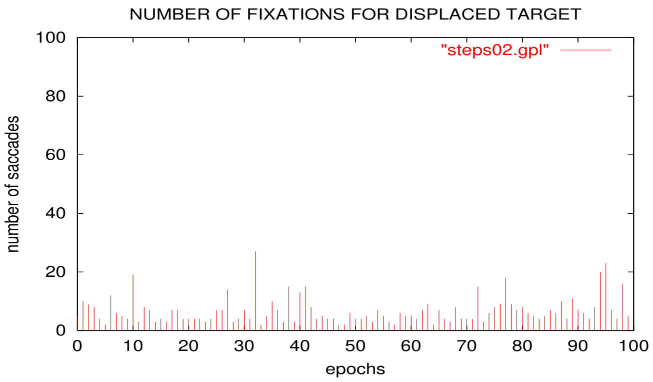


Figure 11: Number of saccades per epoch in finding the object inverted, displaced by 60cm from the training position. Testing from pan=500, tilt=−100.

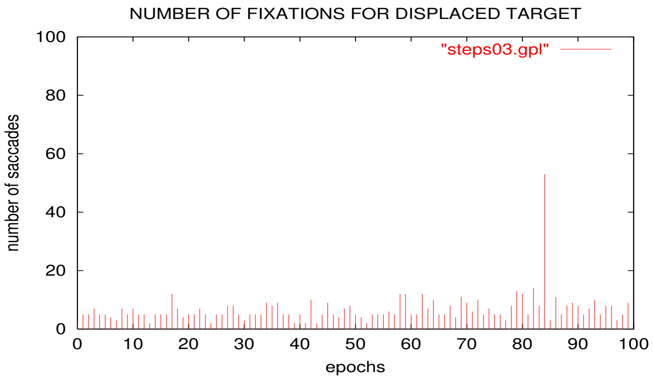


Figure 12: Number of saccades per epoch in finding the object inverted, displaced by 60cm from the training position. Testing from pan=200, tilt=0.

7. CONCLUSIONS AND FUTURE WORK

In the context of active vision we develop a model of selective attention for a visual search task. Our model is a combination of bottom-up visual processing and top-down control for visual attention. Top-down control is achieved by means of reinforcement learning over a low level, visual mechanism of selecting the next fixation, by specifying the gaze direction. Color and symmetry are two low level features that can be used in the selection of the next fixation point. It is not necessary to combine them in a unique

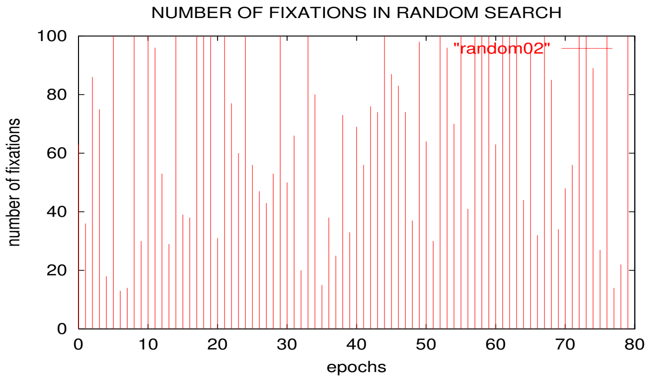


Figure 13: Number of saccades per epochs in a random search, testing from (500, -100). Average: 66.9

saliency map, but rather they can be used sequentially, with greater importance on the former, since color is our feature of choice in recognizing the object. The information that is integrated ("remembered") from saccade to saccade is a coarse signature of the region under the current fixation (the color histogram) and the direction toward the goal region.

The overall performance of our gaze control system can be improved in a number of directions. The Q-learning method can be improved by using *eligibility traces* to remember regions visited previously. The histogram representation of images produces a feature space in which the clusters of images are not completely separated. We can also improve this classifier to produce better separated clusters. We do not need a perfect classifier, because if the system is correct, say, 80-90% of the times, then statistically it will get rewarded more for taking the right actions.

Our model cannot recognize categories of objects (e.g. "desk", "pen", "monitor"), and consequently, cannot learn relationships of the type "the pen is usually on a desk". In the current model, the region where the target object is found does not have semantic meaning. We also need to improve the low-level visual routines to help selecting the fixation points near the center of objects. Other feature maps (e.g. color opponency maps, edge orientation maps, or intensity contrast maps) could be used for a more careful selection of the next fixation point.

Our goal is to extend this approach to develop a gaze control system that can be used on a mobile robot. Here, the problem of visual search is more challenging as the position, and consequently the appearance, of the object can change with the robot's position. In this case, the appearance of the target object changes as the robot moves, so a single template will probably not be sufficient. The distribution of colors in the target object, however, will not change significantly with the point of view, so we expect that the histogram intersection recognizer will still work, even if the robot moves, provided the matching is done at several scales.

Throughout, we have assumed that the environment is sufficiently rich in objects (of various colors), so that we did not have to deal with perceptual aliasing. Extention to a mobile robot, will inevitably lead to learning in *inherently* perceptually aliased environments (e.g. hallways in a building), in which case, no visual information might be sufficient to classify a new image into one of the already learned clusters. We see this as a much more challanging extention. In this case, we will have to resort to using memory to disambiguate the perceptually aliased states, using techniques similar to the ones described in [9].

Acknowledgements

This research is supported in part by a Knowledge and Distributed Intelligence (KDI) grant ECS-9873531 from the National Science Foundation. The computational experiments were carried out in the Autonomous Agents laboratory in the Department of Computer Science and Engineering. We thank the members of the SIGMA group, in particular John Henderson and Fred Dyer, for many illuminating discussions.

8. REFERENCES

[1] S. M. Anstis. A chart demonstrating variations in acuity with retinal position. *Vision Research*, (14):589–592, 1974.

[2] D. H. Ballard. Animate vision. *Artificial Intelligence*, (48):57–86, 1991.

[3] Bandera C., Vico F., Bravo J., Harmon M., and Baird L. Residual q-learning applied to visual attention. In *Proceedings of the 13-th International Conference on Machine Learning, July 3-6, Bari, Italy*, pages 20–27, 1996.

[4] T. Darrel and A. Pentland. Active gesture recognition using partially observable markov decision processes. *13-th IEEE Intl. Conference on Pattern Recognition, Vienna, Austria*, 1996.

[5] J. M. Henderson and A. Hollingworth. Eye movements during scene viewing: An overview. In *Eye Guidance in Reading and Scene Perception*. Elsevier Science B. V., 1998.

[6] L. Itti and C. Koch. A saliency-based search mechanism for overt and covert shifts of visual attention. *Vision Research*, (40):1489–1506, 2000.

[7] 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.

[8] C. Koch and S. Ullman. Shifts in selective visual attention: Towards the underlying neural circuitry. In *Human Neurobiology*. Springer-Verlag, 1985.

[9] A. K. McCallum. *Reinforcement Learning wuth Selective Perception and Hidden State*. PhD thesis, University of Rochester, Rochester, NY, 1996.

[10] M. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley & Sons, 1994.

[11] D. Reisfeld, H. Wolfson, and Y. Yeshurun. Context free attentional operators: The generalized symmetry transform. *International Journal of Computer Vision*, (14):119–130, 1995.

[12] R.S. Sutton and A.G. Barto. *Reinforcement Learning*. MIT Press, 1998.

[13] M. J. Swain and D. H. Ballard. Object identification using color cues. Technical report, University of Rochester, Rochester, NY, 1990.

[14] 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.

[15] C. Watkins. *Learning From Delayed Rewards*. PhD thesis, King's Colledge, 1989.