2009 IEEE International Conference on Robotics and Automation
Kobe International Conference Center
Kobe, Japan, May 12-17, 2009

Learning Sequential Visual Attention Control through Dynamic State Space Discretization

Ali Borji, Majid N. Ahmadabadi, and Babak N. Araabi

***Abstract*— Similar to humans and primates, artificial creatures like robots are limited in terms of allocation of their resources to huge sensory and perceptual information. Serial processing mechanisms used in the design of such creatures demands engineering attentional control mechanisms. In this paper, we present a new algorithm for learning top-down sequential visual attention control for agents acting in interactive environments. Our method is based on the key idea, that attention can be learned best in concert with visual representations through automatic construction and discretization of the visual state space. The tree representing the top-down attention is incrementally refined whenever aliasing occurs by selecting the most appropriate saccadic direction. The proposed approach is evaluated on action-based object recognition and urban navigation tasks, where obtained results support applicability and usefulness of developed saccade movement method for robotics.**

I. Introduction

Huge sensory space, limited response time, dynamicity of the perceptual space and the environment, accuracy and reliability of sensors cause a bounded rationality for a robotic agent. Attention enhances the rationality of the agent by implementing a bottleneck which allows only relevant information to pass to higher level cognitive components like object recognition, scene interpretation, decision making and memory for further processing.

From a large body of existing literature in neuroscience including neurophysiology, psychophysics and modeling studies, it is now known that attention is controlled both by a bottom-up, fast, open-loop, automatic and objective mechanism and also a top-down mechanism which is late, task-driven, closed-loop and subjective. In visual modality bottom-up component is solely determined by low-level image characteristics like color and luminance and is mainly processed by the early visual areas. On the other hand, top-down attention is influenced by the task demands, emotions, expectations, etc. which mainly come from higher cognitive brain areas like prefrontal cortex, LIP, etc [1]. Interactions of these two mechanisms control our attentional behavior. Although we have a good understanding of the bottom-up mechanisms by a broad range of behavioral and modeling studies, to date much less is known in essence and neural mechanisms of the top-down component of visual attention. It is also believed that visual attention is selective to both spatial locations and objects [2]. Not only individual spatial regions are important in terms of the value of information they convey, but also their relationships are significant in deriving attention [3]. Visual attention can also be directed to particular features such as color, orientation and direction of motion [4].

Hand design of control strategies assumes embedding predefined representations in the brain of a mobile robot. Although applicable, such methods are prone to impose overheads in computational complexity and response time. Actually an agent situated in an environment might be able to discover simpler and more efficient representations interactively while having the same efficiency in performing the same tasks. Shaping visual representations dynamically helps the agent to automatically adapt himself to new environmental conditions.

Learning, decision making and attention control are interleaved cognitive processes. It has been shown that eye movements are context-based and task-driven [5]. Previous experiences also influence attentional behaviors which indicate that attention control can be learned [6]. In [7], authors have shown that attention is also affected by decision behaviors. Other studies have proposed that our brain may follow a need-based approach for representing the desired scenes or objects [8]. Considering the above information, semi-supervised approaches in AI and specially RL techniques seem to be the most appropriate tools for learning visual representations and attention control mechanisms. The agent must learn an optimal decision policy along with how to control its attention while interacting with the environment. Such a learning mechanism should help the agent to discover and adopt attention control strategies which are suitable for its needs in a dynamic, complicated and nondeterministic environment. In other words, it should enable the artificial agent to learn to control its visual attention purposefully using top-down attentional signals.

The main contribution of this work is to propose a scalable approach for learning top-down and task-based visual attention control in natural interactive environments by dynamically discretizing visual space whenever aliasing occurs. In order to recognize a scene only a few spatial locations are processed instead of the entire image.

This work was supported by the School of Cognitive Sciences, Institute for Research in Fundamental Sciences, Tehran, IRAN.

Ali Borji is with the School of Cognitive Sciences, Institute for Research in Fundamental Sciences, Niavaran Bldg., P.O. Box 19395-5746, Tehran, IRAN (corresponding author. phone: +98 21-22294035; fax: +98 21-22280352; e-mail: borji@ipm.ir).

Majid Nili Ahmadabadi and Babak Nadjar Araabi are both with the School of Electrical and Computer Engineering, University of Tehran and School of Cognitive Sciences, Institute for Research in Fundamental Sciences, Tehran, IRAN (e-mails: {mnili@ut.ac.ir, araabi@ut.ac.ir}).

Particularly, informative visual features are only extracted and examined at the focus of attention. Our method for learning top-down task-driven visual attention control is computationally efficient and biologically plausible.

The rest of this paper is organized as follows. In section two, some related works are reviewed. Our proposed algorithm for saccade learning is proposed in section three. Experiments and results are shown in section four. Finally, section five concludes the paper.

II. RELATED LITERATURE

Reinforcement learning has previously been used for learning visual attention control in robotics especially for applications like robot navigation and localization.

In [9, 10], a 3 step architecture is proposed for an object recognition task. First, it extracts potential focuses of interest (FOI) according to an information theoretic saliency measure. Then it generates some weak object hypotheses by matching the information at the FOI's with codebooks. The final step is done using Q-learning with the goal of finding the best perceptual action according to the search task.

In [11], two approaches are proposed in a robotic platform with neck, eyes and arms for attention control. The first approach is a simple feedforward method which uses back-propagation learning algorithm while the second one uses reinforcement learning and a finite state machine for state space representation. The robot has 3 types of actions: attention shift, visual improvement and haptic improvement. Their results confirm that the second approach generates a better performance in terms of finding previously observed objects even with fewer movements in head and neck and also in attention center shift. In [12], another robotic platform containing articulated stereo-head with 4 degrees of freedom is presented which can select the region of interest, perform attention shift with saccadic movements, build a map out of the environment and update it according to current observation. In [12], attention control has two steps: first, coarse eye movements and then more precise iterative adjustments around the first points. The termination condition of this process is to reach a maximum correlation among what it finds and what it expects.

An RL approach for learning gaze control for a mobile robot performing a visual search task is proposed in [13]. This 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, merely based 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. In the first module, RL learns a policy on a set of regions in the room for reaching the target object, using an objective function which is the expected value of 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.

In [14], a context-based outdoor scene recognition algorithm for mobile robotic applications is proposed based on the idea of "gist" on top of the saliency model [15]. This method is claimed to have low computational complexity while being biologically plausible.

In [16], Jodogne et al. have proposed an approach for learning action-based image classification known as Reinforcement Learning of Visual Classes (RLVC). RLVC consists of two interleaved learning processes. An RL unit which learns image to action mappings and an image classifier which incrementally learns to distinguish visual classes. RLVC is a feature based attention method in which the entire image is processed to find out whether a specific visual feature exists or not in order to move in a binary decision tree. Like RLVC, our approach also extends the U-TREE [17] to visual domain. The main idea behind the U-TREE algorithm is that aliased states are incrementally refined. Our approach tackles this weakness of RLVC, the exhaustive search for a SIFT feature over the entire image, by computing and searching SIFTs at few locations.

III. PROPOSED LEARNING METHOD

In our method, attention is directed toward spatial circular regions. An attention tree (saccade tree or S-TREE) is incrementally built from the incoming visual inputs. In each node of the saccade tree, visual content at the focus of attention (FOA) is inspected. Visual content in our method is the class of the nearest SIFT feature to the center of FOA. Before learning, features are clustered into some groups.

A. Clustering Local Descriptors (SIFTs)

Sequential attention in our method shifts the focus of attention toward the most informative visual regions. Final output of the algorithm is a scanpath of eye movements for each image. There is no need to represent the pattern at each FOA in fine detail, but an approximate characterization suffices to discriminate among objects and scenes. FOA is a circular region with fixed radius *r*. In order to derive a rough local descriptor representation, SIFT features [18] of some random images $Q = \{q_1, q_2, \dots, q_{|Q|}\}$ were extracted and then clustered using standard *k*-means algorithm. Therefore a set of $|T|$ clusters $T = \{\tau_1, \tau_2, \dots, \tau_{|T|}\}$, hereafter referred to as codebooks, were generated. At each FOA, the codebook, *d*, of the nearest SIFT feature, *k*, is considered as the visual content at FOA:

$$d = argmin_j \; |\mathrm{SIFT}_{FOA,k} - \tau_j|, \quad k \text{ is index of the nearest SIFT to center of FOA} \qquad (1)$$

Fig. 1 shows SIFT features of sample objects from the COIL100 object dataset, used in experiment I. Agent can saccade to one of the eight directions relative to the end of the current position as shown in fig. 1b.

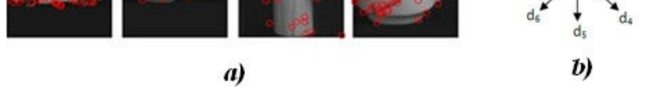

Fig. 1. *a*) Sample objects from the COIL100 dataset with extracted SIFT features. *b*) Discretization of angle into eight saccadic directions with fixed saccade length for encoding of the saccadic movements ($\Delta\alpha = 45^o$).

B. Learning Saccade Tree (S-TREE)

An efficient way to implement attention and state space construction is by means of *tree* data structures. Such structures are interesting because they allow simultaneous learning of representations and attention control.

Visual discretization is performed via attention tree or saccade tree (S-TREE) whenever aliasing occurs. Such refinement is performed to increase cumulative reward of the agent. Each internal node of the S-TREE proposes a single saccade toward one of the eight possible directions. Edges below each node test the codebook seen at the end of the saccade vector in a circular area with radius *r* (FOA). First saccadic position is selected in random and is the same for all images (e.g., center of the image). Based on the observed codebook at the end of the saccade, next saccade is initiated until a leaf node is reached in the tree. Leaves of the tree point to states in the Q-table. Since tree is constructed in a greedy manner, it is prone to overfitting. Solutions should be anticipated to overcome this problem by either periodic tree restructuring or pruning, for instance by merging the nodes with the same best actions or replacing nodes with all their leaves having the same best actions.

While traversing the tree some cases might happen when a codebook seen at the end of the saccade does not exist below a node. In such cases a new child is created with this codebook as its edge label. Such nodes might be noises therefore after some steps they are checked to be removed or finalized based on the number of patterns beneath them.

S-TREE is incrementally built in a quasi-static manner in two phases: 1) *RL-fixed* phase and 2) *Tree-fixed* phase. The algorithm starts with one node in the Tree-fixed phase. In each phase of the algorithm external feedback of the critic, in the form of a scalar reward or punishment, is used to alternatively update the Q-table or refine the attention tree. Initially a tree with a single node is created and all images are mapped to that node. Evidently, such a single state is not enough and aliasing occurs. Then, the algorithm breaks the node to a number of leaves based on some gathered experiences under the node. In each Tree-fixed phase, RL algorithm is executed for a number of episodes by following an ε-greedy or soft-max action selection policy. In this phase, tree is hold fixed and the derived quadruples (s_t, a_t, r_{t+1}, s_{t+1}) are only used for Q-table update according to Q-learning update rule (2).

$$Q(s_t, a_t) = (1-\alpha)Q(s_t, a_t) + \alpha(r_t + \gamma\, max_a\, Q(s_{t+1}, a)) \quad (2)$$

Attention control and state discretization occur in the RL-fixed phase. An important point here is that the agent only accesses the environment through its visual sensor (e.g. its CCD camera), therefore in order to determine its state, it has to traverse its saccade tree from the root node down to a leaf, which determines its state s_t at time t. In the current state, the agent performs an action according to its learned policy. At this point, based on the received reward and the next captured image, which leads to state s_{t+1}, the agent updates the Q value for the state s_t.

After each RL-fixed phase, memory items of leaf nodes are deleted. Leaf nodes without memory are removed from the tree too.

C. Measuring Aliasing

In RL-fixed phase, learned tree is modified to refine leaves with perceptual aliasing. In order to estimate a measure of aliasing, some experiences should be accumulated under a leaf node. This is done by the agent performing some episodes running the current policy learned at the previous Tree-fixed phase. An image is captured, saccade tree is traversed (*traverseTree()* function in table I) in order to find the perceptual state, appropriate action is performed and a reward is received. A good measure of perceptual aliasing in a state (leaf node) is the TD error (Δ_t) and can be derived from the Q-learning formula as in (3).

$$\begin{aligned} Q(s_t, a_t) &= (1-\alpha)Q(s_t, a_t) + \alpha(r_{t+1} + \gamma\, max_a\, Q(s_{t+1}, a)) \\ &= \alpha\left(r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\right) + Q(s_t, a_t) \\ &= \alpha\Delta_t + Q(s_t, a_t) \end{aligned} \quad (3)$$

In order to detect aliasing, all patterns under a node are clustered according to their physical actions and then if any of these clusters has variance in Δ_t's greater than a threshold (*aliasingThreshold*), then that node has aliasing at least with respect to one action. Therefore, Δ_t reduces to (4), because the third term of Δ_t in (3), is the same for all clustered patterns under a node.

$$\Delta_t = r_{t+1} + \gamma\, max_a\, Q(s_{t+1}, a) \quad (4)$$

Δ_t's converge to zero as the RL algorithm converges when there is no further perceptual aliasing. Therefore, in each step of RL, Δ_t is a measure of perceptual aliasing in a state s with respect to an action *a* (*checkAliasing()* in table I).

D. Tree Refinement

When an aliased class is detected, a saccade direction should be selected in order to maximally separate patterns under this class. In order to find the best saccade direction an anticipatory mechanism is needed. When an image ends in state s_t, codebooks along the next eight saccade directions, Δ_t measure of aliasing and performed action are saved as a memory item (*gatherMem()*in table I).

Whenever size of the memory under a state passes a threshold (*maxThreshold*) and it has aliasing, then tree is refined in order to remove aliasing. Tree refinement is then done by selecting the saccade direction which mostly reduces the variance in Δ_t of patterns in memory according to (5).

$$\begin{aligned} [p^* \;\; a^*] &= argmin_{p,a}\left(var(L) - \sum_{c=1}^{|T|}\frac{|L_{a,p,c}|}{|L_a|}var(L_{a,p,c})\right) \\ &= argmax_{p,a}\left(\sum_{c=1}^{|T|}\frac{|L_{a,p,c}|}{|L_a|}var(L_{a,p,c})\right) \end{aligned} \quad (5)$$

In the above formula, L is the set of Δ_t's of all items in memory, L_a is the set of Δ_t's of memory items with action *a*. $L_{a,p,c}$ is the set of Δ_t's of items with action *a*, direction *p* and codebook *c*. $|U|$ and var(U) are the size and variance of the set U. p^* and a^* are the saccadic direction and the action which reduce variance the most, respectively. The whole process of learning saccade tree is summarized in the pseudocode of table I.

TABLE I
PSEUDOCODE OF THE S-TREE ALGORITHM

```
main (void)
 tree = Create a tree with a single node
 Repeat

    for  i = 1 to maxEpizodes          // RL- fixed phase
        $I_t$  = take an image
        $s_t$ = traverseTree(tree, $I_t$)
            [$s_{t+1}$ $r_{t+1}$ $a_t$] = selectAction($s_t$)
            $\Delta_t$ = calcDelta ($s_t$, $s_{t+1}$, $r_{t+1}$, $a_t$)
            mem = gatherMem($a_t$, $I_t$, $\Delta_t$)
    end
    for j=1 to size(tree.leaves)        // Refining states with aliasing
      if  size(mem($s_j$)) >  memTreshold
          if  checkAliasing($s_j$)
              tree = modifyTree(tree, $s_j$)
    end

    pruneTree(tree)                    // Periodic tree pruning

    for  i = 1 to maxEpizodes          // Tree- fixed phase
        $I_t$ = take an image
        $s_t$ = traverseTree(tree, $I_t$)
            [$s_{t+1}$  $r_{t+1}$  $a_t$] = selectAction($s_t$)
            Q-table = updatePolicy(Q-table, $s_t$)
    end

  Until (no more aliasing) or (maximum iterations is reached)
```

```
traverseTree (node,$I_t$)
   if node.childSize > 0
      codebook = getCodeBook(node, I)
      child =  findChild(node, codebook)
      node = traverseTree (child, $I_t$)
   end
   return node
```

```
checkAliasing($s_t$)
  for action $a \in A$                 // A is the set of physical actions
      mem($a$) = all memory items with action  $a$  under $s_t$
      var($a$) = calcVariance (mem($a$))
      if  var($a$) > aliasingThreshold
              return true;
      end
  return false;
```

```
modifyTree(tree, $s_t$)
   for action $a \in A$            // A  is the set of all actions
      mem($a$) = all memory items with action  $a$  under $s_t$
      for  direction $d \in D$       // D  is the set of saccade directions
          calc the conditional variance in (5) and choose the direction
          which reduces variance the most and then break the leaf node
          into a number of new leaves. Also delete the previous state
          associated with this leaf and create new states for new leaves in
          the Q-table
      end
   end
```

IV. EXPERIMENTAL RESULTS

In order to evaluate our algorithm, we have applied it to two visual navigation tasks which capture the main characteristics of real world scenarios. The first task is navigation in a visual gridworld with obstacles and a goal state and the second one is an urban navigation task. In both experiments, 5 SIFT clusters derived from sample images from that database were used to encode the visual contents at FOA's.

A. Navigation in the Visual Gridworld

The aim in this task is to reach the goal state in the upper right corner of the grid shown with red G. The gent has a set of 4 physical actions: move up, right, left and down and has no access to its position in the grid. Agent's only perception of the world is through an image of the object underneath his foot. Any movement taking the agent to an obstacle cell or outside the gird brings it a -1 punishment. When it reaches the goal state, it is rewarded a +1 signal. Each cell of the grid is carpeted with a 128×128 image of the COIL 100 object database which is available at [19]. As shown in fig. 2, S-TREE has managed to recognize all the objects as well as a valid policy by creating 7 distinct perceptual classes after two turns.

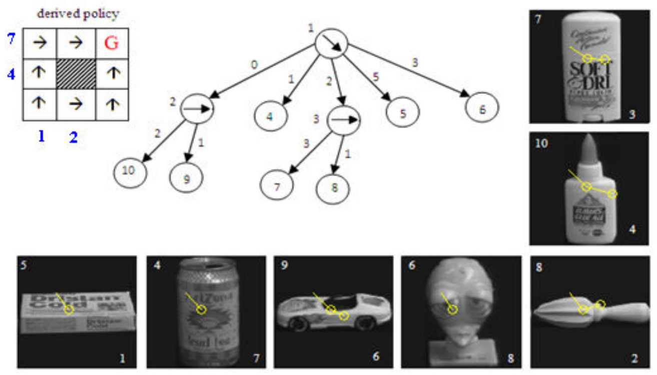

Fig. 2. Performance of the S-TREE algorithm on a 3×3 visual navigation gridworld. The derived policy is shown at the upper-left panel. Numbers in the upper-left and the bottom-right corners of the images determine labels of the tree leaves and grid positions respectively. *aliasingThreshold* was set to 0.3 and size of the saccade vector length was set to 20 pixels. maxEpizodes was 200. α and γ were both 0.9. Radius of the circle at the FOA was 10. *maxThreshold* was 40. Arrows inside the internal nodes and numbers on edges are saccadic directions and codebooks, respectively.

When same objects are assigned to two different locations in the grid, bottom-left and upper-left positions in fig. 3, S-TREE derives 6 states. That is because the best actions for these two positions are the same and therefore there is no need for further refinement. This clearly shows how action-based scene classification differs from ordinary scene classification methods. The latter case also resulted in the optimal policy.

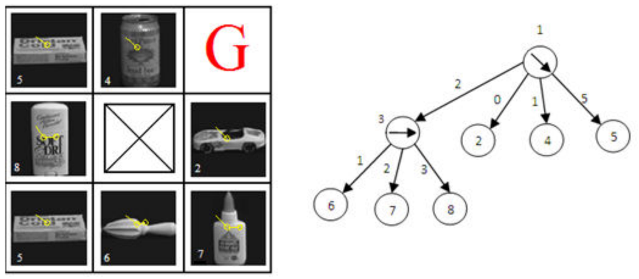

Fig. 3. Generated attention tree (right) for the 3×3 visual gridworld at the left. Parameters of the S-TREE were the same as in fig. 2.

Experimenting with another more complex 6×6 gridworld, shown in figure 4, S-TREE succeeded to derive the optimal policy after 10 iterations (phases). Number of generated visual classes was 28 equal to the number of the non-obstacle non-goal positions in the grid.

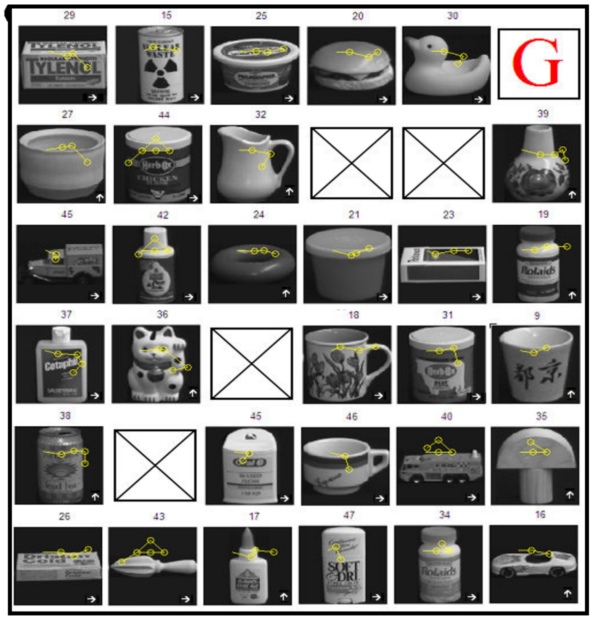

Fig. 4. The 6×6 visual gridworld in experiment I. maxEpizodes was 400. Other parameters were the same as in fig. 2. Learned best actions are shown in bottom-right of the images.

B. Invariancy Analysis

To evaluate the generalization capacity of the S-TREE, we performed an experiment over a 4×4 gridworld with the difference that in each cell, the agent can take a random view of an object. From the total of 16 grid positions, two cells were obstacles and one state was goal. Thirteen objects from COIL100 dataset and 5 views from each one in the range between 0^o and 20^o degrees are assigned to grid positions: therefore, 13×5 = 65 states are possible.

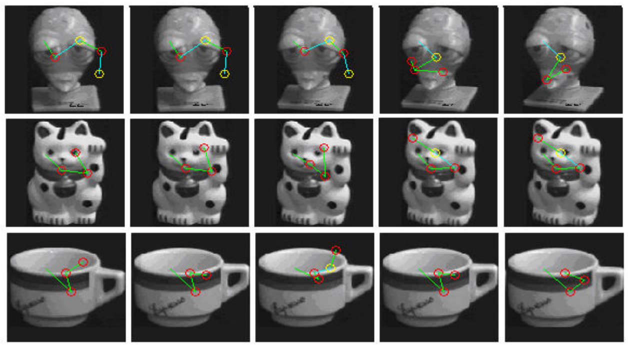

Fig. 5. Parameters for this experiment were as follows: maxEpizodes = 40, iterations = 26, *aliasingThreshold* = 0.6. Other parameters were the same as in fig. 2.

S-TREE succeeded to find an optimal policy for this grid in 16 iterations. Generated S-TREE in this case had 31 perceptual states. This number is greater than the number of gird cells and still less than the possible perceptual states. This shows that S-TREE is to some extent invariant to in-depth rotations. Optimal policy of 100% was achieved. As shown in fig. 5, scanpath of eye movements remained the same for some views with minor rotations.

Fig. 6 shows the distribution of the object views in the perceptual classes for each object. As can be seen, some views of an object are classified in the same perceptual state while others are classified under other states.

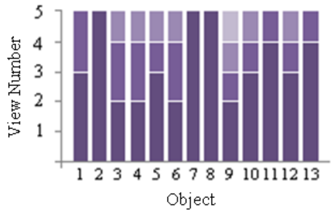

Fig. 6. Distribution of object views in perceptual states.

C. Urban Navigation

We applied the S-TREE algorithm to an urban navigation task, in which the agent is faced with images of natural scenes. This task demonstrates the interactive environments in which the agent has to derive visual states as well as their associated physical actions. The way the agent discriminates the scenes is by looking at special spatial positions. The dataset used in this experiment is freely available at [20].

Fig. 7 shows the map of the environment. This Map consists of 11 locations. Agent's body can orient in 4 possible directions: North, South, West and East. The agent can perform three physical actions: turn left, turn right and go forward and has to enter a goal state which brings it a reward of +100. Each turn brings a punishment of -5 for the agent. Moving forward induces a penalty of -10. In each possible location and head direction, the agent captures a 1024×768 image. The agent must learn to associate scenes to physical actions in order to achieve the task. That way, agent has total number of 11×4 states. S-TREE generated 44 perceptual classes in 24 episodes. Optimal policy was also learned. Generated saccadic eye movements for some sample images are shown in fig. 8.

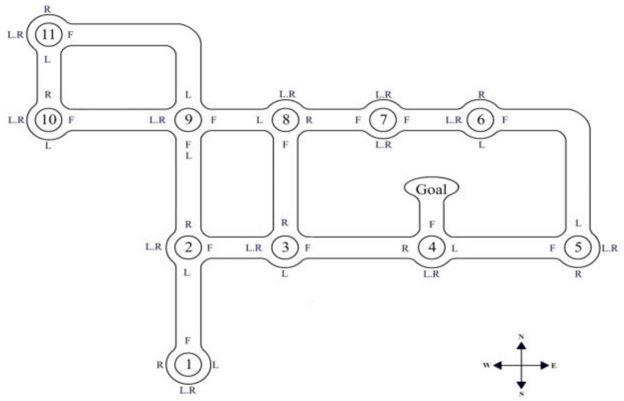

Fig. 7. Map of the environment in the experiment II with the optimal policy. Saccade vector length=150, maxEpizodes=60, *aliasingThreshold*=0.4, r=40, *memThreshold*=30, iterations=12, $\alpha = 0.8$, $\gamma = 0.9$.

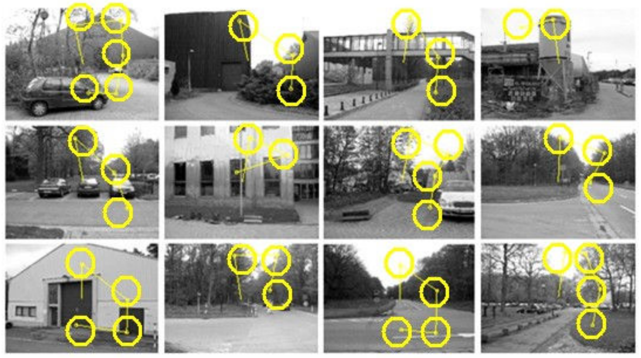

Fig. 8. Sample scenes with learned saccadic eye movements.

Average reward per episode in Tree-fixed phases, number of tree nodes and leaves, average depth of the tree and the number of checks for aliasing in RL-fixed iterations are shown in fig. 9. It shows the number of times patterns in nodes exceeded *memThreshold* and thus demanded checking aliasing. Final generated tree had 62 nodes with 44 of them being leaf nodes and the final average depth of 3.0930. This means that instead of extracting SIFT features in a 1024×768 area they were only calculated at a $3.0930 \times \pi \times 40^2$ pixels area on average.

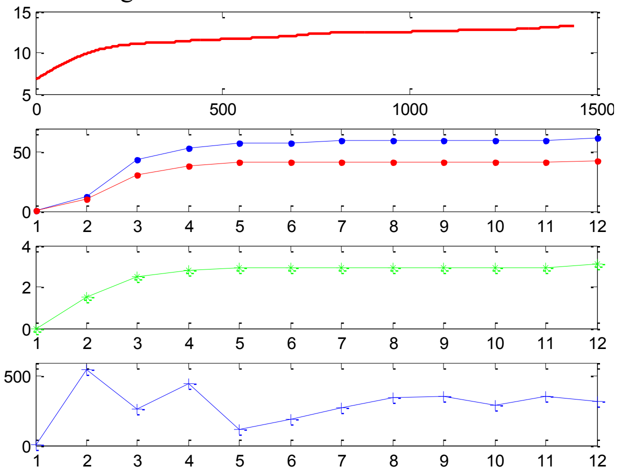

Fig 9. Smoothed average reward (W = 50), number of tree nodes and leaves, average tree depth and checks for aliasing in the second experiment.

V. Conclusions and Discussions

A method for learning top-down visual attention control in interactive environments was presented through dynamically discretizing the visual state space. Compared with the approach in [16], S-Tree algorithm produces trees with smaller depth, making it more suitable for real time applications.

The main contribution of our algorithm is in generating saccade trees which only needs calculating the SIFT features at few spatial locations of the image. It is also in accordance with some behavioral attention mechanisms and is categorized under the space based models of attention. Actually, saliency of the image locations is determined by the expected reward they convey in an interactive task.

Obtained results point toward the fact that active perception together with active movements may play an important role in the mechanisms that the brain uses for representing the world. Therefore in this study we avoided using detailed complex camera-like representations for the environment and instead incorporated attentions and motor actions to form the internal representations. An interesting observation is that, representations are learned interactively and are expanded or shrank adaptively based on the agent's needs. They are also as compact as possible and encode the information at a necessary level without unnecessary details. For instance, for recognizing a scene, it would be very efficient and conclusive to attend to important spatial locations. Therefore global image representation approaches although might propose more accurate solutions in some cases, seem not to be the best solutions where information bottlenecks exist.

In accordance with these views, our method discretizes the visual world when it is needed and when it helps the agent to perform more accurately by removing perceptual aliasing.

It remains to investigate other visual features and saliency maps [15] as candidate locations for the S-TREE algorithm. Such approaches might help to merge need-based capability of the S-TREE with the bottom-up cues of visual attention and may make the algorithm more robust to image transformations. Also extending this approach to continuous action spaces and concurrent clustering of saccadic perceptions along with previous physical actions in a single tree helps the agent to act in POMDP environments.

References

[1] M. Corbetta, and G. L. Shulman, "Control of goal-directed and stimulus-driven attention in the brain," *Nature Reviews Neuroscience*, vol. 3, no.3, pp. 201-215, 2002.

[2] M. I. Posner, and Y. Cohen, "Components of visual orienting," in *Attention and Performance X*, edited by Bouma H and Bouwhuis D. Hillsdale: Erlbaum, pp, 531–556. 1984.

[3] N. Endo, and Y. Takeda, "Selective learning of spatial configuration and object identity in visual search," *Perception & Psychophysics*, vol. 66, no. 2, pp.293-302, 2004.

[4] J. H. Maunsell, and S. Treue, "Feature-based attention in visual cortex," *Trends in Neurosciences*, vol. 29, pp. 317-322, 2006.

[5] A. L. Yarbus, "Eye movements during perception of complex objects," in *Eye Movements and Vision*, ed. L. A. Riggs, Plenum Press, New York, ch. 7, pp. 171–196. 1967.

[6] V. Maljkovic, and K. Nakayama, "Priming of pop-out: I. Role of features," *Mem. & Cognition*, vol. 22, pp. 657-672, 1994.

[7] W. D. Gray, (Ed.), "Integrated models of cognitive systems," New York: Oxford University Press, 2007.

[8] J. Triesch, D. H. Ballard, M. M. Hayhoe and B. T. Sullivan, "What you see is what you need," *Journal of Vision*, vol. 3, no. 86–94, 2003.

[9] G. Fritz, C. Seifert, L. Paletta, and H. Bischof, "Attentive Object Detection Using an Information Theoretic Saliency Measure", *WAPCV*, pp. 29-41, 2004.

[10] L. Paletta, G. Fritz, and C. Seifert, "Cascaded Sequential Attention for Object Recognition with Informative Local Descriptors and Q-learning of Grouping Strategies", *CVPR* 2005.

[11] L. M. G. Gonic, G. A. Giraldi, A. AF. Oliveira, and P.A. Grupen, "Learning Policies for Attentional Control," *IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pp. 294 – 299, 1999.

[12] L. M. G. Gonic, A. Antonio, A. AF. Oliveira, and P.A. Grupen, "A Framework for Attention and Object Categorization Using a Stereo Head Robot," *XII Brazilian Symposium on Computer Graphics and Image Processing*, pp. 143-152, 1999.

[13] S. Minut, and S. Mahadevan, "A Reinforcement Learning Model of Selective Visual Attention," *Fifth International Conference on Autonomous Agents, Montreal*, 2001.

[14] C. Siagian and L. Itti, "Rapid biologically-inspired scene classification using features shared with visual attention," *IEEE Trans. PAMI*, vol. 29, no. 2, pp. 300-312, 2007.

[15] L. Itti, C. Koch, E. Niebur, "A Model of Saliency-Based Visual Attention for Rapid Scene Analysis, *IEEE Trans. PAMI*, vol. 20, no. 11, pp. 1254-1259, 1998.

[16] S. Jodogne, and J. H. Piater, "Closed-Loop Learning of Visual Control Policies," *Journal of Artificial Intelligence Research*, vol. 28, pp. 349-391, 2007.

[17] A. McCallum, "Reinforcement learning with selective attention and hidden state," *PhD thesis, Computer Science Dept, Univ. of Rochester*. 1995.

[18] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[19] http://www.cs.columbia.edu/CAVE/coil-100.html.

[20] http://www.montefiore.ulg.ac.be/~jodogne/phd-database.