

# On-line Learning for Covert and Overt Perceptual Capability for Vision-based Navigation

Zhengping Ji, Xiao Huang, Wei Tong and Juyang Weng  
*Department of Computer Science and Engineering  
Michigan State University  
East Lansing, MI, 48824*

**Abstract**—In this paper, we propose a model to develop robots’ covert and overt perceptual capabilities using reinforcement and supervised learning. Covert perceptual behaviors are treated as actions selected by a motivational system. Overt perceptual behaviors can be learned using either supervised or reinforcement learning. We apply this model to the vision-based navigation system. Instead of dealing with problems in controlled environments with a low-dimensional state space, we test the model on images captured in non-stationary environments. Incremental Hierarchical Discriminant Regression is used to generate states on the fly. Its coarse-to-fine tree structure guarantees real-time retrieval in self-generated high-dimensional state space. K-Nearest Neighbor strategy is adopted to further reduce training time complexity.

**Index Terms**—Reinforcement learning, supervised learning, outdoor navigation, vision.

## I. INTRODUCTION

Supervised (action-imposed) learning is an effective learning mode to help a robot develop intelligent capabilities. However, in psychology and neuroscience [1] [2], there are a lot of cognitive capabilities that can not be developed using this learning mode. For example, visual attention and thinking are two internal behaviors, which cannot be imposed from outside but develop through interactions with the environment. We call the covert behavior, the behavior of a robot that is not visible from the outside. Behaviors that can be imposed directly from the external environment are called overt behaviors.

In this paper, we propose an integrated method that deals with covert and overt perceptual behaviors jointly, using reinforcement and supervised learning. This is the first work that we are aware of that integrates supervised and reinforcement learning for vision-based recognition. In the framework of sensor-driven self-generated representation, this integrated methodology is useful for open-ended learning by a developmental robot. We concentrated on vision-based navigation in everyday natural environments as a challenging task example. This paradigm is important and indispensable from two aspects. First, supervised (action-imposed) learning is not enough to model sophisticated robotic cognitive development since the progress is too tedious and requires intensive human attention. Second, if the robot learns wrong actions it cannot make a correction with this learning mode. On the other hand,

reinforcement learning is a good model to learn intelligent capabilities when imposed action is not acceptable. Moreover, it gives the robot an ability to recover from error, which cannot be gained through supervised learning.

There are quite a few studies which apply reinforcement learning to vision and attention problems. Whitehead and Ballard’s study is one of the earliest attempts to use reinforcement learning as a model of active perception [3]. Bandera [4] used reinforcement learning to solve gaze control problems. Minut and Mahadevan propose a reinforcement learning model of selective visual attention [5]. The work presented in this paper is unique in the following senses: (a) All the internal representation (e.g., clusters that represent distributed states) is generated automatically (sensor-driven) without a need for a human programmer to pre-design task-specific (symbolic) representation; (b) Reinforcement learning and supervised learning are integrated into the same interactive learning system; (c) Instead of using long delay of rewards in traditional reinforcement learning, we use short reward delays only, in order to solve the fast changing perception problems here. To reach the goal of real-time active vision development in non-stationary environments (outdoor navigation), we use IHDR (Incremental Hierarchical Discriminant Regression) [6] to self-generate continuous state space on the fly for high-dimensional visual input. Its coarse-to-fine structure guarantees online retrieval. Furthermore, we implement the k-Nearest Neighbor algorithm for Q-learning so that at each time instant multiple similar states can be updated, which dramatically reduces the number of rewards that human teachers have to issue [7]. We also use a multi-layer learning architecture [8]. The first level learning module handles covert perceptual capability development while the second level learning module deals with navigation behaviors.

In what follows, we first describe the covert capability development problem in outdoor navigation. The detailed architecture of the developmental paradigm is presented in Sec. 3. The experimental results and conclusions are reported in Secs. 4 and 5, respectively.

## II. PROBLEM DESCRIPTION

Our goal is to enable a robot to develop its covert and overt perceptual capability for vision-based outdoor navigation. A

good example is shown in Fig. 1. The left and right color images were captured from a platform of an autonomous driving vehicle. The platform will be discussed later in the experiment section. We used a visor on the vehicle to shadow

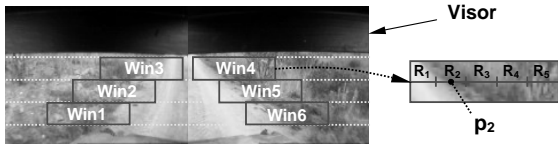


Fig. 1. A pair of images in vision based outdoor navigation with 6 attention windows.

the sunlight that could possibly create bad effects on the image quality. As a result, there is a black top in the image. In addition, some of the bottom part of the image is not in use either because some of vehicle shape is captured in that area. So, in Fig. 1, we only consider the image between the top dash line and bottom dash line. Instead of using the entire part of the image as input to the first-level learning module, we divide two input images into 6 sub-windows (specified by rectangles). Suppose the size of the original image is  $160 \times 80 \times 3$ . The dimension of the state vector is 38400, which is almost intractable. With six windows, the dimension of state vector is reduced to  $80 \times 20 \times 3 = 4800$ . In this way, we reduce the dimension of the state space. And, it turns out, these six windows can cover most of the road boundaries. Our goal is to teach the robot to learn the types of road boundaries. Let us look at the scaled sub-window image in Fig. 1. We divided the centerline of the window into five ranges, which are defined as 5 road boundary types ( $R_1, R_2, R_3, R_4, R_5$ ). In this example, the road boundary intersects the centerline at the point  $p_2$ . Thus,  $R_2$  is regarded as the expected output for this window. In practice, however, it is possible that the boundary may not be in the windows. While the road boundary may be on the left side and the right side of a window, we choose  $R_1$  and  $R_5$  as the expected output, respectively. Although in these situations, the expected output are not exactly fits the road boundaries, it will not take much effect on the decision of final output of heading directions.

Suppose the image of the window is  $x(t)$  (a long vector and each element corresponding to a pixel). The robot needs to learn the mapping from  $x(t)$  to the correct road boundary type  $a = \{R_1, R_2, R_3, R_4, R_5\}$ . The middle point of the correct road boundary type (range) is used as the boundary location for the window image. Since each window has only one intersecting point, we can generate a vector  $E = (e_1, e_2, \dots, e_6)$ . This vector can be the input to the second level module for heading direction learning.

The overall system operation architecture for robotic cognitive development is shown in Fig. 2. A major part of the developmental learning paradigm is IHDR (Incremental Hierarchical Discriminant Regression) [6]. There are IHDR

trees at two levels. At the first level, 6 IHDR trees learn the mapping from visual input to the road boundary types. The second-level IHDR tree maps the road boundary locations of all 6 windows and the desired path to the correct heading direction. The desired path is a spline decided by 10 global position coordinate points. Each coordinate includes the values of longitude, latitude and height. It provides the coarse information of road shape which is derived from maps or other resources. Thus, for the second level, the input is, in total, a 36-dimension vector. The manner in which the robot learns each mapping is discussed in the next section.

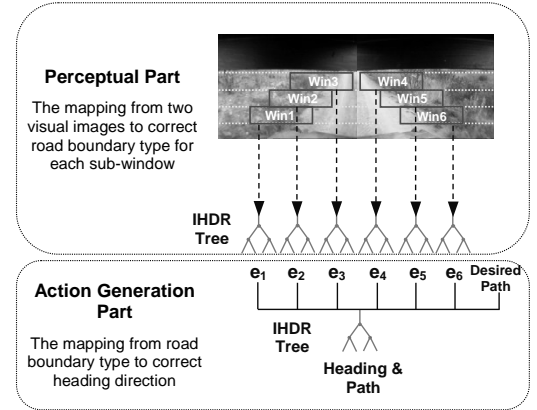


Fig. 2. The system operation architecture for robotic cognitive development.

### III. SYSTEM ARCHITECTURE OF ROBOTIC COGNITIVE DEVELOPMENT

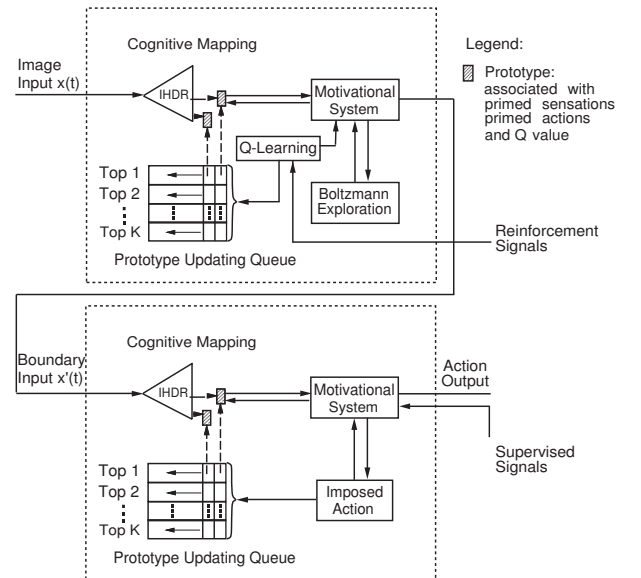


Fig. 3. The system architecture of each learning module for robotic cognitive development.

The basic architecture for robotic cognitive development is shown in Fig. 3. The sensory input is represented by a high dimensional vector. At the first level, We used reinforcement learning. A human teacher will issue rewards and punishments based on the robot's action. In this case, the robot develops this covert capability via internal but non-enforceable probes. For the second level, however, we used supervised learning to give the imposed action directly, since not only sensory input but also the training task are much less complicated than the first level. A human teacher can easily handle the training cases in this level using imposed action. Also, this may save some time cost when considered as a realtime system. In what follows, we will explain each component of the architecture in details.

#### A. Incremental Hierarchical Discriminant Regression

The sensory input  $x(t)$  is fed into the cognitive mapping module realized by IHDR. IHDR clusters  $x(t)$  into current state  $s(t)$  and then maps the current state to the corresponding action output. Thus, IHDR generates state for high-dimensional visual input on the fly, which is a very difficult problem. That is why most studies only deal with low-dimensional state space. In contrast, IHDR makes it possible to handle high-dimensional state space. In the testing phase, given a state  $s(t)$ , the IHDR finds the best matched  $s'$  associated with a list of primed contexts ( $c' = (x', a', q)$ ), which include: primed sensations  $X' = (x'_1, x'_2, \dots, x'_n)$ , primed actions  $A' = (a'_1, a'_2, \dots, a'_n)$  and corresponding Q-values  $Q = (q_1, q_2, \dots, q_n)$ , where  $n$  is the number of different actions. In other words, the function of IHDR is  $g: S \mapsto X' \times A' \times Q$ . The probability to take each primed action is based on its Q-value.

#### B. Q-Learning and Boltzmann Softmax Exploration

There are two major problems in reinforcement learning. First, the reward  $r$  is not always consistent. Humans may make mistakes in giving rewards, and thus, the relationship between an action and the actual reward is not always certain. The second problem is the delayed reward problem. The reward due to an action is typically delayed since the effect of an action is not known until some time after the action is complete. Q-learning [9] can deal with these two problems very well. The basic idea of Q-learning is as follows. At each state  $s$ , keep a Q-value ( $Q(s, c')$ ) for every possible primed context  $c'$ . The action with the largest value will be selected as output and then a reward  $r(t+1)$  will be received. We implemented a modified Q-learning algorithm as follows:

$$Q(s(t), c'(t)) \leftarrow (1 - \alpha)Q(s(t), c'(t)) + \alpha(r(t+1) + \gamma Q(s(t+1), c'(t+1))) \quad (1)$$

where  $\alpha$  and  $\gamma$  are the step-size parameter and the discount parameter, respectively.

The motivational system of the developmental learning paradigm works as an action selection function  $v: 2^{A'} \mapsto A$

( $2^{A'}$  denotes all the possible subsets of  $A'$ ), which chooses an action from a list of primed actions. At the beginning, the robot chooses random actions. Human teachers issue rewards and punishments based on its actions. Through this reinforcement learning procedure, the robot develops the capability to learn road types without imposed actions. In order to let the robot fully explore the action space, Boltzmann Softmax exploration is implemented. At each state ( $s$ ), the robot has a list of actions  $A(s) = (a'_1, a'_2, \dots, a'_n)$  to choose from. The probability for action  $a$  to be chosen at  $s$  is:

$$p(s, a') = \frac{e^{\frac{Q(s, a')}{\tau}}}{\sum_{a' \in A(s)} e^{\frac{Q(s, a')}{\tau}}} \quad (2)$$

where  $\tau$  is a positive parameter called temperature. With a high temperature, all actions in  $A(s)$  have almost the same probability to be chosen. When  $\tau \rightarrow 0$ , the Boltzmann Softmax exploration most likely chooses action  $a$  that has a high Q-value.

#### C. Prototype Updating Queue

In the batch learning mode of a Q-learning algorithm, back-propagation is applied to all states. For real-time development, this global iteration method is not applicable due to the excessive time required. We must use a local method that only involves a small number of computations that go through a local state trajectory. Thus in Fig. 3, we designed the prototype updating queue to store the addresses of formerly visited states. At each time step, after the sensory input is received, the primed sensation is updated according to the following expression:

$$x^{(n)}(t) = x^{(n-1)}(t) + \frac{1+l}{n} \gamma (x(t-1) - x^{(n-1)}(t)) \quad (3)$$

where  $l$  is the amnesic parameter. If  $l > 1$ , it means the latest input contributes more.

Thus, not only is the Q-value back-propagated, so is the primed sensation. This back-up is performed iteratively from the tail of the queue back to the head of the queue. After the entire queue is updated, the current state's address is pushed into the queue and the oldest state at the head is pushed out of the queue. Because we can limit the length of the queue, real-time updating becomes possible.

#### D. K-Nearest Neighbor Q-Learning

The state space would continue to grow as the robot explores in new environments. In order to reduce the training time, we adopt the k-Nearest Neighbor strategy. For each state, its top k nearest neighbors are saved in the prototype updating queue (Fig. 3). If a reward is issued, the system not only updates the current state-action pair but also updates the k-NN state-pairs, which dramatically reduces time complexity of the training procedure. Suppose the state space is  $S$  and the current state is  $s(t)$ . Let us define  $D$  as the

distance between  $s(t)$  and the  $k$ th nearest neighbor of  $s(t)$ . The volume of the set of all states whose distance from  $s(t)$  is less than  $D$  is defined as follows:

$$A(k, S, s(t)) = \frac{2D^n \pi^{n/2}}{n\Gamma(n/2)}. \quad (4)$$

Using the k-NN updating rule, the updated space goes from one point  $s(t)$  to  $A$ , which is a tremendous improvement.

We can also look into the training complexity issue. Suppose for each state we need to issue  $m$  rewards to get a reliable training result (Q-value converges) and the number of states is  $N$ , then we have the time complexity of training formulated as:

$$T_{1-NN} = mN \quad (5)$$

With top-k updating, we assume each state has the same possibility to be chosen as a match. The time complexity is

$$T_{k-NN} = N + \frac{(m-1)N}{K} \quad (6)$$

where the first  $N$  means that each state has to be visited at least once while  $\frac{(m-1)N}{K}$  is the number of training sessions needed using top-K updating. The ratio of time complexity using top-1 updating and top-k updating is

$$\frac{T_{k-NN}}{T_{1-NN}} = \frac{N + \frac{(m-1)N}{K}}{mN} = \frac{k+m-1}{mk} \quad (7)$$

Even though the assumption of the equal probability of each state to be chosen as top match is too strong, we still can see the potential improvement of top-k updating.

#### E. Algorithm

The algorithm of the developmental learning paradigm works in the following way:

- 1) Grab a new sensory input  $x(t)$  and feed it into the IHDR tree. The IHDR tree generates a state  $s(t)$  for  $x(t)$ .
- 2) Query the IHDR tree and get a matched state  $s'$  and related list of primed contexts.
- 3) If  $s(t)$  is significantly different from  $s'$ , it is considered as a new state and the IHDR tree is updated by saving  $s(t)$ . Otherwise, use  $s(t)$  to update  $s'$  through incremental averaging.
- 4) In the first level, using the Boltzmann Softmax Exploration in Eq. 2, chose an action based on the Q-value of every primed action. In the second level, give the imposed action directly. Execute the action.
- 5) Based on the retrieval action, give a reward.
- 6) Update the Q-value of states in PUQ using k-NN updating rule. Go to step 1.

### IV. EXPERIMENTAL RESULTS

We trained the robot to learn road boundary types and heading direction for vision-based navigation and compared k-NN Q-learning to traditional Q-learning.

#### A. Q-Learning

First, we will show the teaching result of learned road boundary type for one image (state). In total, there are 5 possible actions ( $a = \{R_i | i = 0...4\}$ ), in which  $R_1$  is the correct action. Fig. 4 plots the Q-value of each action from visit No. 1 to visit No. 30. After training with reward and punishment information, the Q-value of each action converges to the specific value. From the figure we can see the Q-value of Action 1 ( $R_1$ ) is the largest (positive) while the Q-values of other actions converge to negative values. The learning process takes approximately 17 visits to converge.

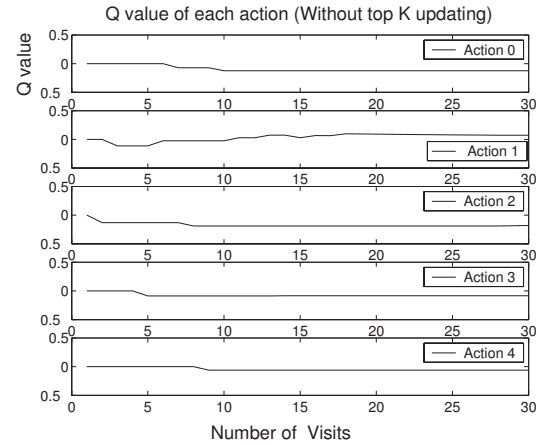


Fig. 4. Q-value of each action(top-1 Q-learning).

In the next experiment, we trained 50 consecutive images repeatedly for 20 times. The strategy of top 5 nearest neighbors updating is used here. We should notice that by using the top K updating mechanism, the number of updates ( $n_u$ ) is different from the number of visits ( $n_v$ ).  $n_v$  means the number of times that state  $s$  is retrieved as a top 1 match while  $n_u$  means the number of times that state  $s$  is retrieved as one of the top K matches.

The Q-value of each action versus the number of updates is shown in Fig. 5. Here "+" means that the state is chosen as top 1 match. Action 1 should be the correct action and presents the largest value (positive) after some training. Because of the trainer's mistake at the beginning, Action 0 got several positive rewards (Q-value increases) while Action 1 got a couple of negative rewards (Q-value decreases). However, after 30 times of updating, Action 1 has the largest value and the system automatically chooses it as the output. So, the plot also verifies that k-NN-based reinforcement learning is able to tolerate human mistakes.

Because we used Boltzmann exploration to choose the action, in Fig. 6, the first plot is generated to show the probability of each action based on its Q-value. The probabilities of action 0, 1, 2, 3 and 4 are plotted from bottom to top,

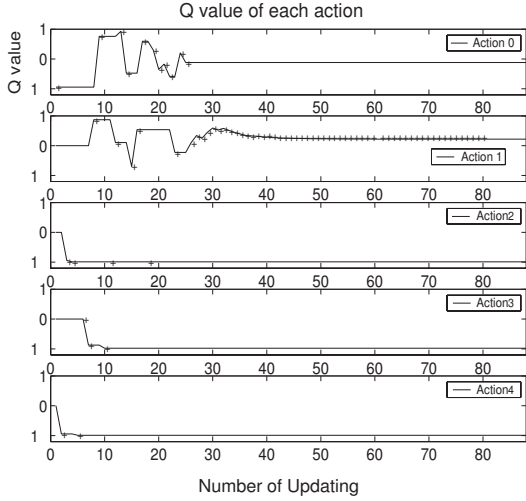


Fig. 5. Q-value of each action (k-NN Q-learning).

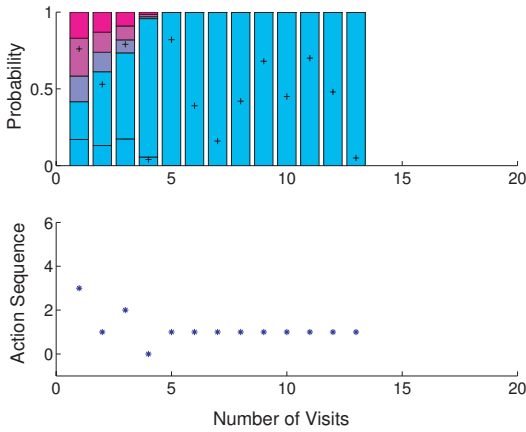


Fig. 6. The first plot shows Boltzmann Softmax Exploration. The second plot shows the action sequence.

respectively. The "+" denotes the random value generated by a uniform distribution. If the random value is in one range, say, the bottom range, then Action 0 would be taken. Only information from visit No. 1 to visit No. 15 is shown here. As we can see, at the beginning, each action has a similar probability (about 0.2). After training, Action 1 is chosen most of the time. The action sequence is shown in the second plot of Fig. 6. When  $n_v > 4$ , the system only chose action 1. That is, for each attention point it takes about 5 visits to converge. This again shows the advantage of k-NN-based reinforcement learning.

### B. Retrieval Testing

The retrieval time of each image is shown in Fig. 7. The average training time of each step is 0.0186s and the highest training time is 0.12s. Even though we have 6 trees for 6 windows, the system can still work in real time. This shows

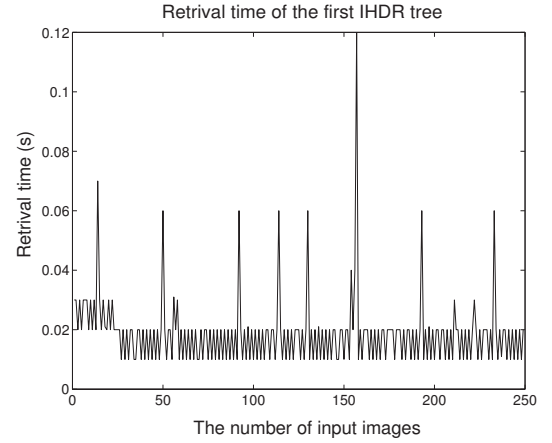


Fig. 7. Retrieval time of each image for one tree.

how efficient IHDR is for robot mental development.

### C. Experiments with Autonomous Driving Vehicle

In this experiment, we developed the covert perceptual capability of our autonomous driving vehicle, which was fabricated by Autonomous Vehicle Systems Corp. and Embodied Intelligence Laboratory at Michigan State University. The main architecture of the platform is shown in Fig. 8. We will only discuss the vision system here, by using the algorithms in the last section.

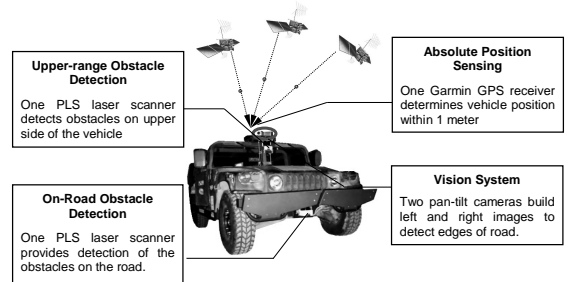


Fig. 8. The Platform of Autonomous Driving Vehicle

Fig. 9 shows the interface used for the experiment of road boundary detection. We designed key buttons "a", "s" and "c" to issue the rewards "-1", "+1" and "0", respectively. We used the vehicle to capture 200 consecutive road images with a time step of 1 second. The reinforcement learning method discussed in the last section was used to train each sub-window image. For each 5 images, we leave out one image for testing. The remaining images are used for training. Fig. 10 shows the average error rate of detected road boundaries. Here, the rounds of training means that in each round, we will give rewards for each sub-window image of all image data. From the result, we can see that the learning method shows good performance of detection ability after some time of training.

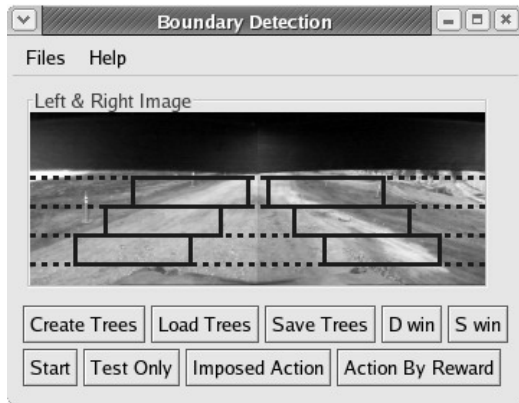


Fig. 9. The interface for road boundary detection (Perceptual Part).

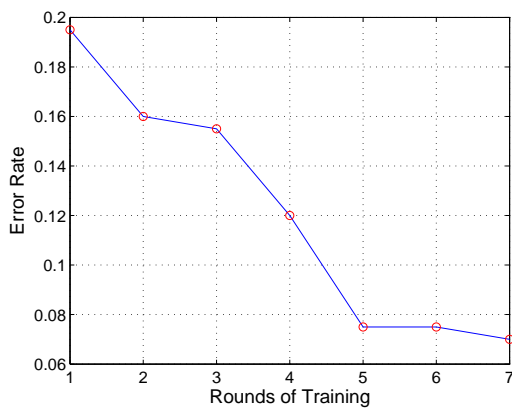


Fig. 10. Error rate for the road boundary detection

Fig. 11 shows the interface used for vision navigation, which is action generation part in our architecture. The six points received from the first level is projected on the actual vision window. The curve drawn on the actual vision window is the desired path decided by 10 global position coordinate points, which is artificially generated here. The primed vision window shows the nearest neighbor from the IHDR tree associated with current heading direction (displayed under the primed window). Imposed action will be given by clicking the mouse to the expected heading direction in the primed vision window. This training session is the supervised learning. The test results are shown in Table. I. In the re-substitution test, we used all the training samples to test, the MSE (Mean Square Error) of the heading direction remained at zero. In the disjoint test, we leave out one sample in each five samples for testing. The other 4 samples were used for training. The heading direction from retrieval result is compared with the heading direction of ground truth. The MSE shows that the robot is able to drive without a big deviation from the correct path we expected.

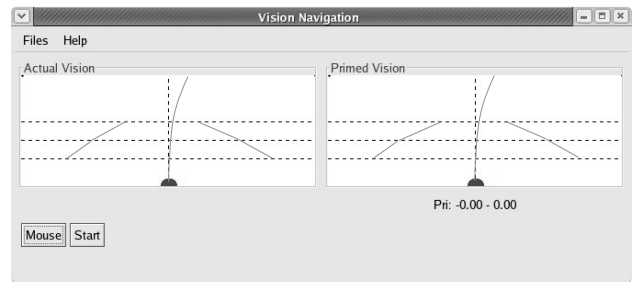


Fig. 11. The interface for vision navigation (Action Generation Part).

TABLE I  
MEAN SQUARE ERROR FOR HEADING DIRECTION

Test Mode	Re-substitution	Disjoint
Mean Square Error	0 °	1.1269 °

## V. CONCLUSIONS AND FUTURE WORK

The work presented here showed that supervised and reinforcement learning modes can be integrated into an open-ended developmental robot, so that the human teacher can take advantage of the two learning modes. Although the ideas of such an integration are intuitive, sophisticated architectures and the associated developmental algorithm makes it possible for on-line learning by generating high-dimensional internal representation automatically. We will extend the model to develop other vision-based navigation capabilities in the future. More real-time experiments will also be done based on the platform of an autonomous driving vehicle.

## REFERENCES

- [1] J. Piaget, *The Origins of Intelligence in Children*, International Universities Press, INC, New York, 1952.
- [2] J. Flavell, P. Miller, and S. Miller, *Cognitive Development*. Prentice-Hall, Englewood Cliffs, NJ, 1993.
- [3] S. Whitehead, and D. Ballard, "Active perception and reinforcement learning," *Neural Computation*, vol. 2, pp. 409-419, 1990.
- [4] C. Bandera, F. Vico, J. Bravo, M. Harmon, and L. Baird, "Residual q-learning applied to visual attention," *Proc. of the 13th Int'l Conf. Machine Learning*, pages 20-27. Bari, Italy, 1996.
- [5] S. Minut, and S. Mahadevan, "A reinforcement learning model of selective visual attention," *Proc. of the 5th Int'l Conf. Autonomous agents*, pp. 457-464. Montreal, Canada, 2001.
- [6] W. Hwang, and J. Weng, "Hierarchical discriminant regression," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1277-1293, 1999.
- [7] X. Huang and J. Weng, "Covert Perceptual Capability Development Using Reinforcement Learning," *Proc. of the 5th Int'l Workshop on Epigenetic Robotics*, pp. 22-24, Nara, Japan, 2005.
- [8] Y. Takahashi, and M. Asada, "Vision-guided behavior acquisition of a mobile robot by multi-layered reinforcement learning," *IEEE/RSJ Int'l Conf. Intelligent Robots and Systems*, pp. 395-402, 2000.
- [9] C. Watkins, "Q-learning," *Machine Learning*, vol. 8, pp. 279-292, 1992.