How To Bring Up A One-Eyed Mobile Robo-Infant

Takayuki Nakamura and Minoru Asada
Dept. of Mechanical Eng. for Computer-Controlled Machinery,
Osaka University, Suita 565 JAPAN
Phone:+81-6-879-7349, Fax:+81-6-879-7348.
e-mail: nakamura@robotics.ccm.eng.osaka-u.ac.jp

Abstract

An animat, a simulated animal or a real robot, has to learn a sequence of actions which reflect the dynamic interplay between the animat and its environment, mediated through the animat's sensors and actuators. This paper presents a method to bring up an animat, a one-eyed mobile robo-infant in such a way that it learns the sensorimotor apparatus first (in its babyhood), and then how to detect and avoid obstacles and how to pursuit a target (in its childhood), and finally how to combine these behaviors (in its youth). Key issues dealt with in this paper are twofold:

- The robo-infant has visual tracking routines capable of realtime acquisition of optical flow (visual motion patterns) by which it can obtain the sensorimotor apparatus, and target pursuit and obstacle avoidance behaviors based on reinforcement learning.
- The system's (parents') direct training (programming or teaching how to do) is avoided. Instead, the system changes the environments so as to make the robo-infant learn several behaviors to accomplish some tasks step by step.

Computer simulation and real experiments are given to show the validity of the method.

1 Introduction

An animat, a simulated animal or a real robot, has to learn a sequence of actions which reflect the dynamic interplay between the animat and its environment, mediated through the animat's sensors and actuators. In order for the animat to be able to learn behaviors, many issues related to each other should be considered.

First, the animat has to find situated cues to associate them with embedded motor commands. In the subsumption architecture [1], these cues are embedded, or more correctly specified by the programmer. As a method of automatically finding these cues, the researchers in robot learning have tried to make agents learn to behave against the perceived data from the external world [2]. The candidates for the cues, however, are selected in advance for behavior learning for the specified task among the numerous number of perceived data. Especially, in the reinforcement learning scheme [3], the state which is the result of some action should be predefined, which means that the candidates for situated cues are involved in the state variables. These cues seem dependent on scene components and limited to the specified situations and the task. Therefore, detection of such cues seems unstable in different real situations.

Visual motion cues can be considered as robust ones independent of scene components and capable of being tightly coupled with motor commands. The visual motion cue such as an optical flow has been used in CV area for 3-D scene reconstruction [4]. Recently, the main trend of CV has been shifting to the purposive active vision paradigm [5, 6, 7]. Further, tight coupling between visual motion cues and motor commands seems important as shown in biology [8] or physiological psychology [9]

Second, the environment has a very important role from a viewpoint of how to bring up a robo-infant. Almost of the all existing methods in robot learning assume the fixed environment for their individual tasks and have not considered to change the environments so as to make the animat learn more complicated behaviors step by step. In [10], we proposed the Learning from Easy Missions (hereafter, LEM) paradigm in which the reinforcement learning time can be reduced from the exponential order into the linear order of the state size by placing the animat near the goal state at the beginning and farther from it later. Although the environment is fixed in this case, the animat could learn the shooting behavior successfully.

In this paper, we present a method to bring up an animat, a one-eyed mobile robo-infant which has visual tracking routines capable of realtime acquisition of optical flow of the environment that are used as visual cues to be associated with several behaviors. We extend the LEM paradigm from only the animat's placement in the fixed environment [10] to changing the environments so that the animat can learn complicated behaviors gradually. At the beginning the animat learns the sensorimotor apparatus in its babyhood with almost no obstacles, and then learns several behaviors such as detecting and avoiding obstacles, and pursuit a target in its childhood, and finally learns to combine these behaviors in its youth.

In the next section, we describe our robo-infant; the system, the actuators, the sensors, and the brain, and then the tasks it tries to accomplish. Next, we give a method for acquisition of the fundamental relationship between visual motion cues and robot motor commands, and describe a reinforcement learning method to obtain target pursuit behavior with computer simulations and real experimental results.

2 The Robo-Infant and Task

(a) System

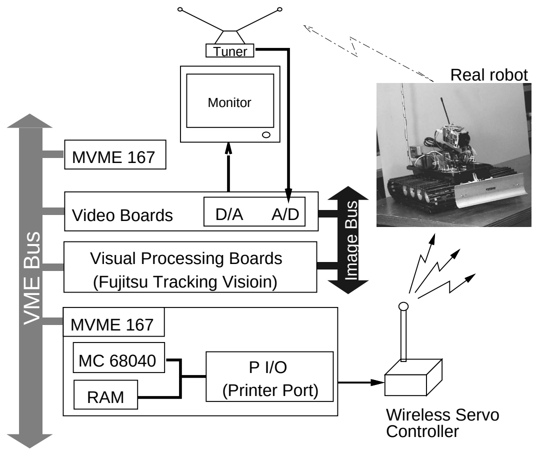
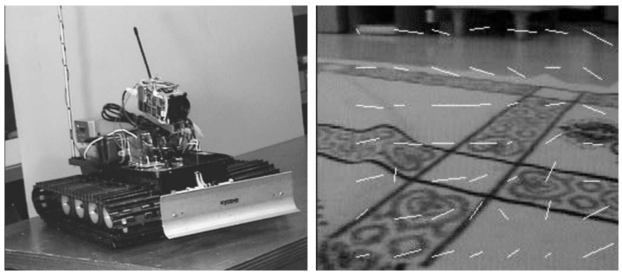


Figure 1: **A configuration of the system.**

Fig.1 shows a configuration of our one-eyed mobile robo-infant. We have constructed the radio control system of the robot[10]. The image processing and the vehicle control system are operated by VxWorks OS on MVME167(MC68040 CPU) computer which are connected with host Sun workstations via Ether net. The image taken by the robo-infant is transmitted to a UHF receiver and subsampled by scan-line convertor(Sony Corp.). Then, the video signal is sent to a Fujitsu tracking module. The tracking module has a function of block correlation to track some pre-memorized patterns and can detect motion vectors in real time. The Datacube MaxVideo 200, a real-time pipeline video image processor is used to synthesized pre-memorized reference block images. Then the tracking module feeds the flow vectors at each regions to the host CPU(MC68040). The host CPU calculates the averaged motion vector field (see Section 3.1 for more detail) and stores them. The host Sun workstation calculates the SVD off-line. **Fig.2** shows a picture of the robo-infant with a TV camera (Sony camera module) and video transmitter.

(b) Actuators

The robo-infant has a Power Wheeled Steering system driven by two motors into each of which we can send a motor command, independently. The velocities of translation v and rotation ω of the robot can be represented by two motor commands, more correctly two anglur velocities ω_l and ω_r. The following equation shows the relationship between (v, ω) and (ω_r, ω_l) to



(a) robo-infant (b) detected image

Figure 2: **A picture of the robo-infant and motion detection.**

be sent to the right and left motors.

$$\begin{pmatrix} v \\ \omega \end{pmatrix} = \begin{pmatrix} \frac{R_r}{2} & \frac{R_l}{2} \\ \frac{R_r}{T} & -\frac{R_l}{T} \end{pmatrix} \begin{pmatrix} \omega_r \\ \omega_l \end{pmatrix} \qquad (1)$$

where R_r, R_l, and T denote the radii of the right and left wheels, and the distance between two wheels, respectively.

In our experment, we quantized $\omega_{l(r)}$ into five levels which correspond to quick forward, slow forward, stop, slow backward, and quick backwrad, respectively. Totally, we have 25 actions. Note that the robot does not even know any physical meanings of these actions.

(c) Sensors

To detect changes due to the robot motion, we use real-time visual tracking routines which can track about 140 windows (each window consists of 8 by 8 pixcels) in real-time (video rate) by using a motion estimation processor (MEP) [11]. Searching area is 16 by 16 pixcels and MEP outputs the location of each window where the following matching error (SAD: sum of absolute difference) is minimum.

$$D[i,j] = \sum_{k=0}^{K-1} \sum_{l=0}^{L-1} |R[k,l] - M[i+k, j+l]|$$

$$i, j : 0 \le i, j \le 15,$$

where $R[x,y]$, $M[x,y]$, and $D[x,y]$ denote a reference block, a matching block, and an array of SAD, respectively. The visual tracking routines are used to obtain an optical flow of the floor, to track a target specified by human operator, and to detect and avoid obstacles.

(d) Brain and Task

Fig.3 shows a basic idea of visual motion guided behavior. The visual motion guided behavior constructed in the brain consists of visual and motor behaviors for each task. The basic component of the visual behavior is a set of visual tracking routines based on template matching method for optical flow detection or target tracking. The image area to be covered by these tracking routines are specified or automatically detected depending on the individual tasks, and the

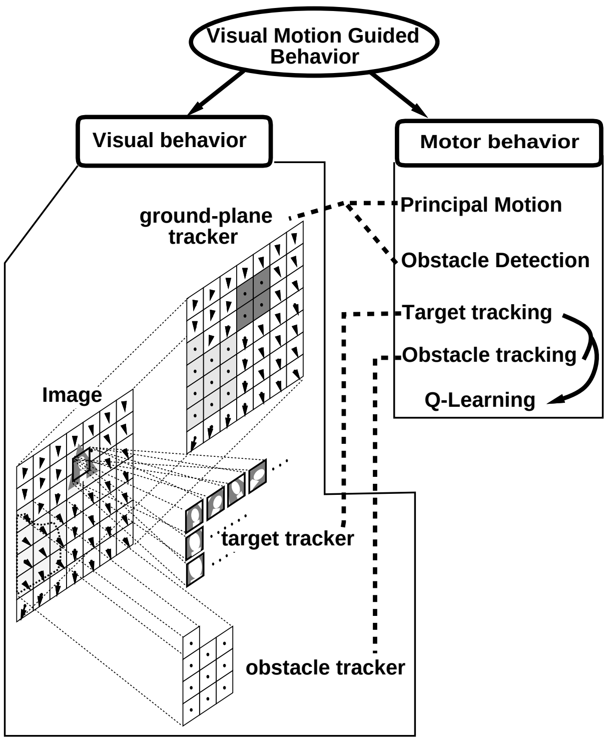


Figure 3: **Visual Motion Guided Behavior**

cooperative behavior between tracking routines are conducted for the task accomplishment.

The most fundamental task is to obtain the relationship between the visual motion and robot motor commands. To do that, the visual tracking routines are scattered over the whole image and an optical flow due to instantaneous robot motion is detected. In this case, the tracking routines are fixed to the individual image positions. In the task of obstacle detection and avoidance, the candidates for obstacle are first detected by comparing the motion vector with that of non-obstacle (ground plane) region, and then this region is tracked by multiple templates each of which tracks the inside of the moving obstacle region. For the target pursuit task, the multiple templates are set up and every template looks for the target for the stable tracking.

The motor behavior is a set of motor commands obtained by Q-learning, one of the robot learning methods, based on detected motion cues and given task. The size and position of the target or the detected obstacle which are successfully tracked are used as a state vector.

The visual motion guided begaviors work in parallel in the image and compose the layered architecture. The visual motion guided behavior for monitoring robot motion (detecting the optical flow on the ground plane on which the robot lies) is the lowest and might be subsumed in part due to occlusion by other visual motion guided behaviors for obstacle detection/avoidance and target pursuits which might occlude each other.

3 Babyhood (Acquisition of Sensorimotor Apparatus)

Here, we show how to learn sensorimotor apparatus by correlating sensor information with motion command. We assume that the robot is given no knowledge of the structure of its sensory system nor of the effects. We extend the method [12] as follows:

- Instead of sonar information (3-D range information), we use optical flow of the floor which can be obtained by multiple visual tracking routines.
- In order to remove fluctuations of flow pattern of each action due to the environmental factors, we set up the environment with almost no obstacles. In averageing flow patterns, we used the least median of squares method [13] to remove the outliers due to noise or small obstacles.

Finally, we can condense the visual motion patterns by the obtained fundamental relationship and then use it to include the ego-motion information in the internal state space of the agent.

3.1 Aquisition of Sensorimotor Apparatus

Here, we call an optical flow pattern due to a particular action a primal motion vector field (hereafter, *pmvf*). We place 49(7×7) visual tracking routines to detect changes in the whole image.

We obtain a mapping that gives the averaged *pmvf* for any particular actions. This mapping is obtained as follows: The space of actions is divided into a set of boxes. In a real system, the robot moves around the floor by a PWS (Power Wheeled Steering) system with two independent motors. We can send the motor control command to each of two motors independently. Two motor commands ω_l and ω_r have 5 subactions (quick forward(qf), slow forward(sf), stop(st), slow backward(sb) and quick backward(qb) motions), respectively. The entire space of actions is thus divided into 25 boxes. These boxes can be specified by two indices where the first component of the index is the right motor command and the other is the left motor command. An averaged *pmvf* is obtained for each of these boxes. Each time the robot takes an action $i(\tau_{li}, \tau_{ri}) \quad \tau_{li}, \tau_{ri} \in \{qf, sf, st, sb, qb\}$, the averaged *pmvf* $\boldsymbol{p}_i$ is updated for the box corresponding to the action just taken. Examples of these vector fields in a real environment are shown in **Fig.4**. In the environment without obstacles, the robot randomly selects a possible action among the action space and executes it. While random wandering, the robot stores the flow patterns for each actions.

3.2 Aquisition of principal motions

Using the mapping from the actions to the averaged *pmvf*s obtained in the last subsection, we acquire principal motions which characterize the space of actions. This is done by analyzing the space of averaged *pmvf* that robot is capable of producing. We want to find a basis for this space, i.e., a set of representative

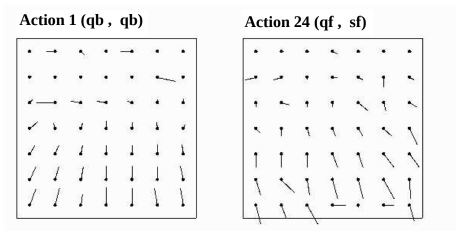


Figure 4: **Examples of averaged *pmvf*s in a real environment.**

motion vector fields from which all the motion vector fields may be produced by linear combination. We can obtain the representative motion vectors by using Principal Component Analysis that may be performed using a technique called Singular Value Decomposition(hereafter SVD). The sample values of $\boldsymbol{p}_i$ (the averaged *pmvf*) corresponding to the actioin $i(\tau_{li}, \tau_{ri})$ are organized as the rows of matrix $\boldsymbol{P}$. There are 25 of these fields each having 98 components (49 vectors per one box). The SVD of $\boldsymbol{P}$ is

$$P_{m\times n} = U_{m\times n} S_{n\times n} E^T_{n\times n}, \tag{2}$$

where S is a diagonal matrix whose elements are the singular values of $\boldsymbol{P}$ and the rows of $\boldsymbol{E}^T$ are the desired orthonormal basis vectors. Here, $m = 25$, the number of averaged *pmvf*s and $n = 98$, the number of components in each averaged *pmvf* (two for each local visual tracking routine). $\boldsymbol{U}$ is the orthogonal matrix in terms of the row. The averaged *pmvf* $\boldsymbol{p}_i$ can be written as a linear combination of the vectors in $\boldsymbol{E}^T$ from the quation (2), using the K principal components:

$$p_i \approx \sum_{k=1}^{K} u_{ik} s_k e_k^T \tag{3}$$

Thus, we have found a basis set (the row vectors of $\boldsymbol{E}^T$) for the space of averaged *pmvf*. In fact, we obtain 26 principal components by caluculating SVD for the $\boldsymbol{P}$ which consist of 25 flow patterns. **Fig.5** shows the singular values with respect to the principal components.

We select two important basis vectors among them which have larger singular value than others. The averaged *pmvf* may be approximated by throwing away all but the important basis vectors. Thus, for example, vector $\boldsymbol{p}_i$ may be approximated by

$$p_i \approx u_{i1} s_1 e_1^T + u_{i2} s_2 e_2^T$$

if we keep only the first two components. The first two principal components obtained in the real environment are shown in **Fig.6**. Clearly, the first (a) corresponds to a pure rotation and the second (b) to a pure backward motion.

Next step is to make a relation between the possible actions $\boldsymbol{p}_i$ by representing each of them in terms of

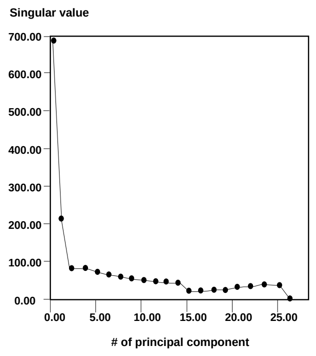
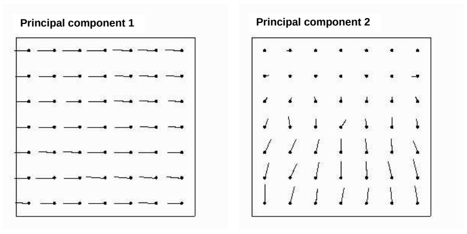


Figure 5: **Singular values of principal components.**



(a) (b)

Figure 6: **The first two principal components.**

the coeficient $a^i_k = u_{ik} s_k$ in the action space which consists of two principal components. The relation between possible actions of the real robot are shown in **Fig.7**, where the number indicates the number of the action i $(i = 1 \sim 25)$.

4 Childhood (Behavior Acquisition Based on Reinforcement Learning)

4.1 Basics of Reinforcement Learning

Reinforcement learning agents improve their performance on tasks using reward and punishment received from their environment. They are distiguished from supervised learning agents in that they have no "teacher" that tells the agent the correct response to a situation when an agent responds poorly. An agent's only feedback indicating its performance on the task at hand is a scalar reward value. One step Q-learning[14] has attracted much attention as an implementation of reinforcement learning because it is derived from dynamic programing[15] and because it works well. Here, we briefly review the basics of the Q-learning. We follow the explanation of the Q learning by Kaelbling [16].

We assume that the robot can discriminate the set $\boldsymbol{S}$ of distinct world states, and can take the set $\boldsymbol{A}$ of actions on the world. The world is modeled as a Markov process, making stochastic transitions based on its current state and the action taken by the robot.

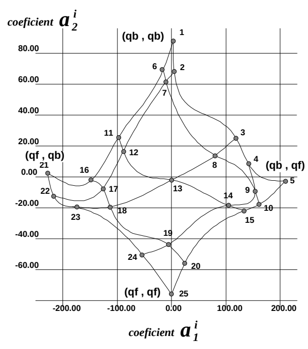


Figure 7: **The relation between the possible actions of the real robot.**

Let $T(s, a, s')$ be the probability that the world will transit to the next state s' from the current state-action pair (s, a). For each state-action pair (s, a), the *reward* $r(s, a)$ is defined.

The general reinforcement learning problem is typically stated as finding a policy that maximizes discounted sum of the reward received over time. A policy f is mapping from $\boldsymbol{S}$ to $\boldsymbol{A}$. This sum called the *return* and is defined as:

$$\sum_{n=0}^{\infty} \gamma^n r_{t+n}, \tag{4}$$

where r_t is the reward received at step t given that the agent started in state s and executed policy f. γ is the discounting factor, it controls to what degree rewards in the distant future affect the total value of a policy and is just slightly less than 1.

Given definitions of the transition probabilities and the reward distribution, we can solve the optimal policy, using methods from dynamic programming [15]. A more interesting case occurs when we wish to simultaneously learn the dynamics of the world and construct the policy. Watkin's Q-learning algorithm gives us an elegant method for doing this.

Let $Q^*(s, a)$ be the expected return or *action-value function* for taking action a in a situation s and continuing thereafter with the optimal policy. It can be recursively defined as:

$$Q^*(s,a) = r(s,a) + \gamma \sum_{s' \in S} T(s,a,s') \max_{a' \in A} Q^*(s', a'). \tag{5}$$

Because we do not know T and r initially, we construct incremental estimates of the Q values on line. Starting with $Q(s, a)$ at any value (usually 0), every time an action is taken, update the Q value as follows:

$$Q(s,a) \Leftarrow (1-\alpha)Q(s,a) + \alpha(r(s,a) + \gamma \max_{a' \in A} Q(s', a')). \tag{6}$$

where r is the actual reward value received for taking action a in a situation s, s' is the next state, and α is a leaning rate (between 0 and 1).

According to the above formalization of the state set, the action set, and other functions and parameters, we apply the Q-learning to target pursuit task.

4.2 Learning Method

We use visual tracking routines in order to pursuit a target specified by an human operator and obtain the information about the target in the image. This information is used in the Q-learning algorithm for aquisition of target pursuit behavior. Here, we show the relation between the state of visual tracking routine and the state used in Q-learning scheme.

4.2.1 Visual functions of tracking routine

Our visual tracking routine have the following visual functions.

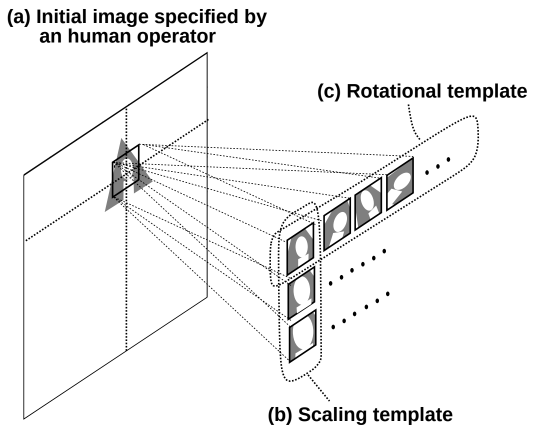


Figure 8: **The target image and the reference block images.**

1. Scalable and rotatable matching: We can generate three level scaling images and eleven level rotational images as reference block images. For example, **Fig.8** shows an inital image specified by an human operator (a), three scaling images (b), and eleven rotational images (c), respectively.

2. When the visual tracking routine tracks an image, it applies the block matching process with the reference block stored in the initial frame and the search window selected to be centered around the last tracking position from the current frame. By this function, we can obtain the target position in the image while pursuiting the target.

3. When the target detection fails, search-whole-image routine is called in order to detect the target again.

We define the state of the target in the image based on the target position and the target size (three levels) obtained by visual tracking routine. We detect motion vector in an image by applying the block matching process with the reference block ($t = t_i$) and the search window images ($t = t_{i+1}$) continuously. Thus, we can obtain the optical flow vectors in the image at any time.

4.2.2 State and action spaces in Q-learning

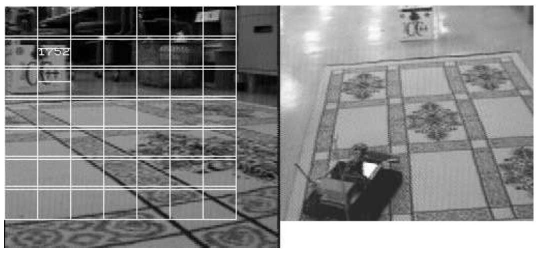
In order to apply the Q-learning scheme to target pursuit task, we define a number of sets and parameters.

A state set $\boldsymbol{S}$: the state of the target in the image is quantized into 9 sub-states, combinations of three positions (left, center, and right) and three sizes (large (near), medium, and small (far)). Similarly, changes in terms of target's position and size in the image are quantized into 9 sub-states, combination of three states in terms of position's change (move left, no move and move right) and three state in terms of size's change (enlarge, no change, shrink). We add the two lost situations (target is lost into the left side or the right side) into the state space. Furthermore, we add the action(totally 25 actions) just taken on observing the current situation into the state space. Totally, we have 2300 states in the set $\boldsymbol{S}$.

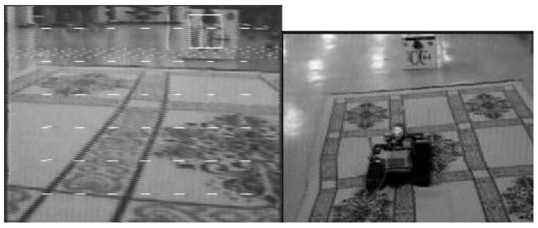
A action set $\boldsymbol{A}$: Totally, we have 25 actions in the action set $\boldsymbol{A}$. We assign a reward value 1 when the robot touched the target or 0 otherwise. A discounting factor γ is used to control to what degree rewards in the distant future affect the total value of a policy. In our case, we set the value a slightly less than 1 ($\gamma = 0.9$).

4.2.3 Target tracking with no obstacles

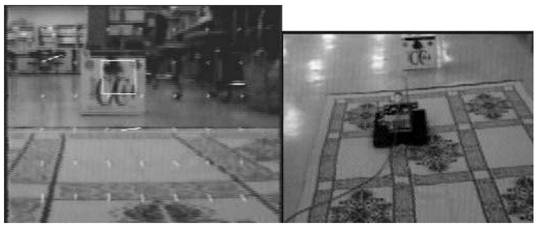
The experiment consists of two phases: first, learning the optimal policy f through the computer simulation, then apply the learned policy to a real situation. The merit of the computer simulation is not only to check the validity of the algorithm but also to save the running cost of the real robot during the learning process. **Fig.9** shows a sequence of images where the robot succeeded in pursuiting a target. The top left figure in **Fig.9** (a) shows the initial position of the target. The top right shows the reference block images which is synthesized for tracking the target. The left figures in the **Fig.9** (b), (c) and (d) shows the processed images. Each white rectangle in these images shows the tracked target position. The white lines shows the optical flow. In this way, based on the hierachical architecture of the visual tracking routine, we can perform the target tracking and the optical flow detection in parallel on the real system.



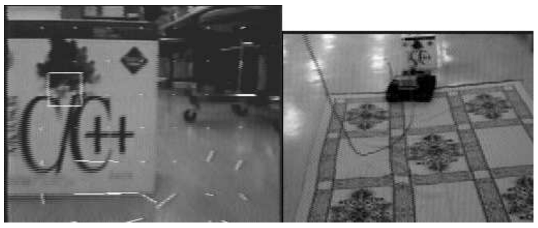
(a)



(b)



(c)



(d)

Figure 9: **The robot succeeded in pursuiting a moving target.**

4.3 Obstacles Detection and Tracking

(a) Detection and tracking of obstacles by flow differences

We know the flow pattern $\boldsymbol{p}_i$ corresponding to the action i in the environment without any obstacle. Motion segmentation is done by comparing the flow pattern $\boldsymbol{p}_i$ with the flow pattern $\boldsymbol{p}_i^{obs}$ which is obtained in the environment with obstacles.

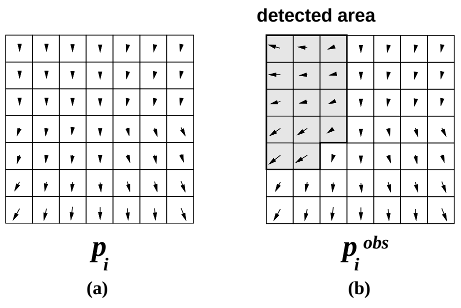


Figure 10: **Obstacle Detection.**

The area in the $\boldsymbol{p}_i^{obs}$ which differs from the same area in the $\boldsymbol{p}_i$ is detected as the area in which the obstacle candidates are projected. This information (position and size in the image) is used to obtain the obstacle tracking behavior.

Fig.11 show the result of obstacle detection in the real environment, where (a) displays the environment in which the robot detected candidates for obstacles (see (b)). The circles in the image indicate the obstacle candidate regions.

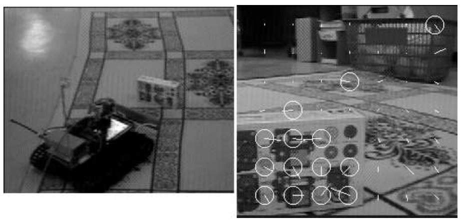
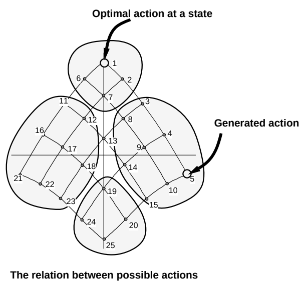


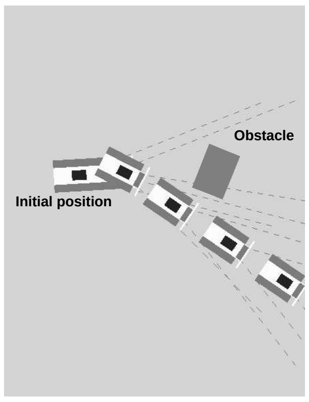
Figure 11: **A picture of the environment with an obstacle and obstacle detection.**

(b) Learning obstacle avoidance behavior

The learning obstacle avoidance consist of two stages. First, the obstacle tracking behavior is learned by the same manner as in learning the target pursuit behavior. Next, the obstacle avoidance behavior is generated by using the relation between the possible actions and the obstacle tracking behavior as follows: (1) the relationship between the possible actions is divided into four categories by clustering the action space represented by the coefficients (a_1^i, a_2^i) (See **Fig.12**(a)), (2) the obstacle tracking behavior is mapped on the relationship, and the category C^twhich includes the obstacle tracking action is found out, (3) the obstacle avoidance actions are selected among the categories excluding the C^t. More correctly, the obstacle avoidance action is obtained by finding the action having the smallest action-value function with respect to the obstacle tracking behavior among the categories except for C^t. **Fig.12**(b) shows the obstacle avoidance behavior acquired by the proposed scheme in computer simulation.


Optimal action at a state
Generated action
The relation between possible actions

(a) Generation of obstacle avoidance behavior



(b) Obstacle avoidance behavior in simulation.

Figure 12: **Obstacle avoidance behavior**

5 Youth (Coordination of Behaviors)

We consider coordinations in which the previously learned behaviors are combined: switching action value functions according to the situation. We used the subsumption architecture [1] to combine previously learned behaviors. The switching condition is to use target pursuit behavior unless only the obstacle can be observed very largely. **Fig.13** shows the results of the coordinated behavior acquired by our method.

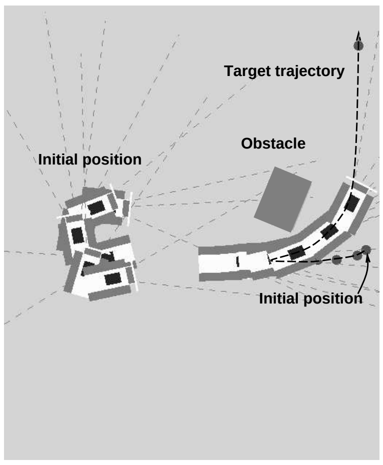


Figure 13: **Coordinated behavior in simulatoin.**

6 Discussions ans Future Work

We proposed a method to bring up a robo-infant by letting it learn from a single behavior in a simple environment to combined behavior in a comlicated environment. As one of the general cues for situated agents to behave against the external world, we utilized visual motion cue which is independent of scene components and tightly coupled with motor commands. Although the cue does not depend on the scene components, the tracking routines for detecting optical flow and pursuiting a target depends on the fluctuations of intensities due to the change of the lighting conditions during the robot actions. Normalization technique of the intensity might be useful but it takes too long time. We must make the tracking routines more robust by applying other techniques.

Due to the limit of the current hardware system, it takes some time to set up the multi templates for obstacle tracking after obstacle detection. Therefore, the current system loses the quickly moving obstacles. Now, we are planning to develop a new program which tightly connects MaxVideo 200 and Fujitsu Tracking module to speed up and add other higher level visual functions.

Since the number of DOFs of the robot motion is currently too few, just two, we have a plan to extend our system by adding an active stereo camera head which has at least four DOFs in order to increase the capability of robot behaviors by visual motion. Furthermore, it is useful to pursuit a target which is often likely to be lost in a fixed camera system.

References

[1] R. A. Brooks. "A robust layered control system for a mobile robot". *IEEE J. Robotics and Automation*, Vol. RA-2, pp. 14–23, 1986.

[2] J. H. Connel and S. Mahadevan, editors. *Robot Learning*. Kluwer Academic Publishers, 1993.

[3] R. S. Sutton. "Special issue on reinforcement learning". In R. S. Sutton(Guest), editor, *Machine Learning*, Vol. 8, pp. –. Kluwer Academic Publishers, 1992.

[4] M. Tarr and M. Black. "Dialogue: A computational and evolutionary persepctive on the role of representation in vision". *CVGIP: Image Understanding*, Vol. 60:1, pp. 65–73, 1994.

[5] Y. Aloimonos. "Reply: What I have learned". *CVGIP: Image Understanding*, Vol. 60:1, pp. 74–85, 1994.

[6] G. Sandini and E. Grosso. "Reply: Why Purposive Vision". *CVGIP: Image Understanding*, Vol. 60:1, pp. 109–112, 1994.

[7] S. Edelman. "Reply: Representatin without Reconstruction". *CVGIP: Image Understanding*, Vol. 60:1, pp. 92–94, 1994.

[8] G. A. Horridge. "The evolution of visual processing and the construction of seeing systems". In *Proc. of Royal Soc. London B230*, pp. 279–292, 1987.

[9] R. Held and A. Hein. "Movement-produced stimulation in the development of visually guided behaviors". *Journal of Comparative and Physiological Psycology*, Vol. 56:5, pp. 872–876, 1963.

[10] M. Asada, S. Noda, S. Tawaratsumida, and K. Hosoda. "Vision-Based Behavior Acquisition For A Shooting Robot By Using A Reinforcement Learning". In *Proc. of IAPR / IEEE Workshop on Visual Behaviors-1994*, pp. 112–118, 1994.

[11] H. Inoue, T. Tachikawa, and M. Inaba. "Robot vision system with a correlation chip for real-time tracking, optical flow and depth map generation". In *Proc. IEEE Int'l Conf. on Robotics and Automation*, pp. 1621–1626, 1992.

[12] D. Pierce and B. Kuipers. "Learning to Explore and Build Maps'. In *Proc. of AAAI'94*, pp. 1264–1271, 1994.

[13] A.Rosenfeld P. Meer, D.Mintz and D.Y.Kim. "Robust Regression Methods for Computer Vision: A Review". *IJCV*, Vol. 6:1, pp. 59–70, 1990.

[14] C. J. C. H. Watkins. *Learning from delayed rewards"*. PhD thesis, King's College, University of Cambridge, May 1989.

[15] R. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

[16] L. P. Kaelbling. "Learning to achieve goals". In *Proc. of IJCAI-93*, pp. 1094–1098, 1993.