Electronics and Communications in Japan, Part 3, Vol. 89, No. 7, 2006
Translated from Denshi Joho Tsushin Gakkai Ronbunshi, Vol. J87-D-II, No. 7, July 2004, pp. 1505–1515

Adaptive Acquisition of Dynamics Matching in Sensory-Motor Fusion System

Naoko Ogawa,[1] Yutaka Sakaguchi,[2] Akio Namiki,[3,1] and Masatoshi Ishikawa[1]

[1]Graduate School of Information Science and Technology, University of Tokyo, Tokyo, 113-8656 Japan

[2]Graduate School of Information Systems, University of Electro-Communications, Chofu, 182-8585 Japan

[3]CREST, Japan Science and Technology Agency, Kawaguchi, 332-0012 Japan

SUMMARY

"Dynamics matching" is a design concept in sensory-motor fusion systems, where the performance is maximized by adjustment of the dynamics of the system under the physical and computational constraints. This paper models dynamics matching as an optimization problem and constructs an adaptive acquisition algorithm. A numerical experiment on the target-tracking task using active vision is presented as an example, and it is shown that a reasonable solution is acquired by the proposed approach. © 2006 Wiley Periodicals, Inc. Electron Comm Jpn Pt 3, 89(7): 19–30, 2006; Published online in Wiley InterScience (www.interscience.wiley. com). DOI 10.1002/ecjc.20255

Key words: sensory-motor fusion; dynamics matching; reinforcement learning; target-tracking; robotics.

1. Introduction

There has recently been remarkable progress in element technology for robotics. With the development of sensors and actuators with high-speed operation, it is expected that tasks can be performed at ultrahigh speed [1–3]. When the operating speed of the system is improved, on the other hand, the effects of the temporal characteristics (dynamics) of various elements can no longer be ignored. In particular, the physical dynamics of the actuator and sensor performances, and the computational dynamics of the computation resources and the algorithm, greatly affect the system performance.

In the control of servomotors, for example, a control rate of at least 1 kHz is generally considered necessary [4]. When sensor information is used in the control of the actuator, there must be a sensor with a corresponding sampling frequency. Conversely, even if high-speed sensors are introduced, their speed is not fully taken advantage of if the actuator is slow. Furthermore, the system performance depends greatly on the algorithm and the information processing strategy of the processing system.

Thus, in order to integrate the sensing, processing, and motor systems of a robot consistently and to achieve maximum performance as a sensory-motor fusion system [5], it is not sufficient to consider only speed improvement of the individual elements: the matching of their dynamical properties should also be considered.

Consequently, Namiki and colleagues proposed the concept of dynamics matching, as a unified framework for the physical and computational dynamics [5, 6]. The concept is that the dynamics of the sensing, processing, and motor systems of the whole system should be adjusted subject to the physical and computational constraints and

matched to the dynamics of the task and the outer world, so that the performance is maximized according to the situation.

Consider, as an example, the situation in which a robot does not have sufficient physical performance in the sense of sensor accuracy or actuator output, and this deficiency is to be compensated by prediction or estimation. Real-time computation has certain limits, however, due to the operating speed of the processor and the capacity of the memory device. Thus, a problem arises due to the following two contradictory requirements.

(1) The computational complexity must be increased in order to compensate for deficiencies of physical performance.

(2) The computational complexity must be reduced in order to maintain real-time operation.

Under the trade-off caused by these two conditions, the information processing strategy and the computational complexity must be controlled dynamically. Dynamics matching is the process of solving such problems by controlling the physical and computational dynamics of the system.

Past studies have not taken a sufficiently unified approach to the physical and computational dynamics, and the solution of the problem has depended on the experience and the intuition of the engineer at the site. In robotics, the physical dynamics of the robot has long been considered in trajectory planning and other problems [7–9], but there has been little explicit discussion of computational resources or computation load. Consequently, the potential capabilities of the hardware and software of robots has not been fully utilized.

In contrast, in the fields of artificial intelligence and management engineering, there have been studies of estimation and the decision-making subject to the limits of given computational resources [10, 11]. In the fields of parallel processing and hard real-time scheduling, resource allocation is a major issue [12, 13]. However, most of these studies do not consider physical dynamics and are not suitable for application to robot systems in the real world. The concept of dynamics matching is one attempt to find a solution of the above problem in system design.

Two levels should be considered in the realization of dynamics matching. One is realization at the stage of system design [5, 6]. However, it is in general difficult to acquire the *a priori* properties of the elements. Neither is it easy to adapt to variations of the properties. The other is to adjust the dynamics matching state adaptively and on-line to changes of situation. This can be achieved by applying an exploratory technique, even if the properties are not known *a priori*.

This paper considers the latter problem of adaptive dynamics matching as an optimization problem, and proposes an adaptive acquisition algorithm. Using the target-tracking task as an example, the algorithm is implemented in a numerical experiment.

2. Algorithm for Adaptive Acquisition of Dynamics Matching

2.1. Formulation of problem

First, consider dynamics of various properties of the whole system, which is composed of the robot system, tasks, and the whole outer world. The properties include physical ones (such as the maximum torque of the motor, the range of motion, the dynamic range of the sensors, and the sensitivity of the sensors), and computational ones (such as computational resources, algorithms, and the decision-making process).

The dynamics of these properties are divided into ones that cannot be adjusted directly by the system and ones that can be adjusted directly. Below, the former are called the constrained dynamics $\boldsymbol{c}$ and the latter are called the adjustable dynamics $\boldsymbol{a}$. The system performance is denoted as P. Then, the dynamics matching problem can be considered as the problem of finding the optimal adjustable dynamics $\boldsymbol{a}$ that maximizes the performance P under the given constrained dynamics $\boldsymbol{c}$.

Figure 1 illustrates the above optimization problem. The bottom surface in the figure is the space spanned by the vectors whose components are the system properties. Its axes consist of the constrained dynamics $\boldsymbol{c}$ and the adjust-

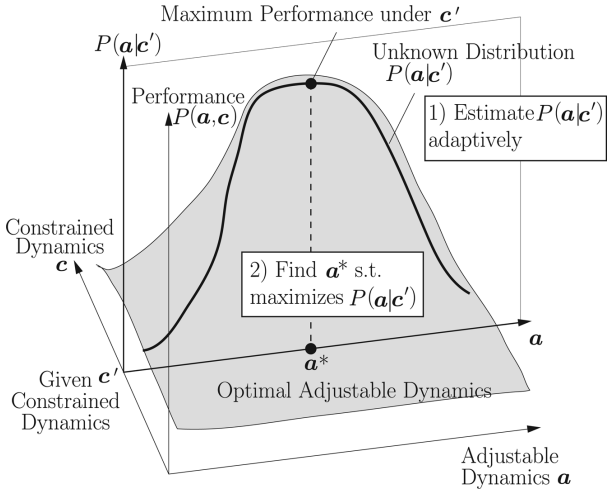


Fig. 1. A model of the dynamics matching problem.

able dynamics $\boldsymbol{a}$ (it is assumed, for simplicity, that there is no interaction among them). The vertical axis is the distribution of the performance P ($\boldsymbol{a}$, $\boldsymbol{c}$), which is generally unknown to the system. When a constrained dynamics $\boldsymbol{c}'$ is given, it implies that the space is cut by a fixed hyperplane $\boldsymbol{c} = \boldsymbol{c}'$. The subspace obtained as the cross section represents the performance distribution P ($\boldsymbol{a}|\boldsymbol{c}'$) under the constrained dynamics $\boldsymbol{c}'$. The purpose of the proposed method is to vary the adjustable dynamics $\boldsymbol{a}$ in this subspace so as to find $\boldsymbol{a}^*$ that maximizes the performance P, that is,

$$\boldsymbol{a}^* = \underset{\boldsymbol{a}}{\operatorname{argmax}}\, P(\boldsymbol{a}|\boldsymbol{c}')$$

This optimization problem is composed of the following two steps.

(1) P ($\boldsymbol{a}|\boldsymbol{c}'$), which is unknown, is to be estimated adaptively.

(2) $\boldsymbol{a}$ that maximizes P ($\boldsymbol{a}|\boldsymbol{c}'$) is to be found.

It is difficult to perform these two steps at the same time by an analytical method, and an exploratory approach is required. Furthermore, modeling of the system is almost impossible in many cases because of the ever-changing dynamics. It is desirable to derive the solution on-line while operating the real machine. Thus, parallel search is almost impossible, and it is required to examine the system behavior separately for each value of the adjustable dynamics.

In order to deal with such a requirement, in this study we attempt to perform a search by reinforcement learning, which has an affinity to robotics, being a trial-and-error approach based on interaction with the real world. Reinforcement learning has been applied successfully to the solution of large-scale and complex problems, such as dynamic channel allocation for cellular telephones [14] and parameter adjustment in fuzzy control [15]. In the next section, dynamics matching is formulated in terms of reinforcement learning, and a solution algorithm is proposed.

2.2. Acquisition algorithm based on reinforcement learning

Reinforcement learning is a trial-and-error learning procedure intended to maximize the reward that can be obtained from the environment [16]. The procedure differs greatly from supervised learning in that learning progresses even if the goal pattern to be acquired is not explicitly presented.

The subject of reinforcement learning is called the agent. The agent observes the "state" of the environment and performs an "action" in accordance with a certain "policy." The environment makes a state transition according to the action and gives a "reward" to the agent, which represents the goodness or badness of the action. Learning proceeds so that the "value function," which is the expectation of the reward, is maximized.

In order to maximize the expectation of the reward, the reward must be predicted. The agent estimates the value function by interaction with the environment and evaluates the state and the action in order to predict the reward. The agent determines the policy on the basis of the value function, and selects the action.

Reinforcement learning with such a feature is applied below to dynamics matching acquisition, as shown in Fig. 2. Consider first the cross section P ($\boldsymbol{a}|\boldsymbol{c}'$) obtained by the cut made by a hyperplane $\boldsymbol{c} = \boldsymbol{c}'$. The bottom face of the hyper-cross section is a (dim $\boldsymbol{a}$)-dimensional space with the adjustable dynamics $\boldsymbol{a}$ as its element (for simplicity, Fig. 2 is drawn for the one-dimensional case). This space is denoted as $\mathcal{P}$.

In the proposed algorithm, the point $\boldsymbol{a}$ (adjustable dynamics) in space $\mathcal{P}$ is considered as the "state" in the reinforcement learning, and the movement $\Delta\boldsymbol{a}$ in space $\mathcal{P}$ is considered as the "action." A certain value which can serve as an index of real-time performance is considered as the "reward." Then, the distribution of the value function V ($\boldsymbol{a}$, $\Delta\boldsymbol{a}$) comes to reflect strongly the performance P ($\boldsymbol{a}|\boldsymbol{c}'$) with the progress of learning.

Then, the purpose of the algorithm is to move around in the state space by trial and error, adjusting the adjustable dynamics, and to find the state that maximizes the performance of the whole task.

It is assumed below, for simplicity, that the state space $\mathcal{P}$ is a discrete space. However, the theory can also be extended to a continuous space. In addition, in this study, the search in the parameter space is considered as an analogy to the gridworld problem [16], and the action is defined as an incrementation of the adjustable dynamics. However, the definition of the action is not the essential aspect of

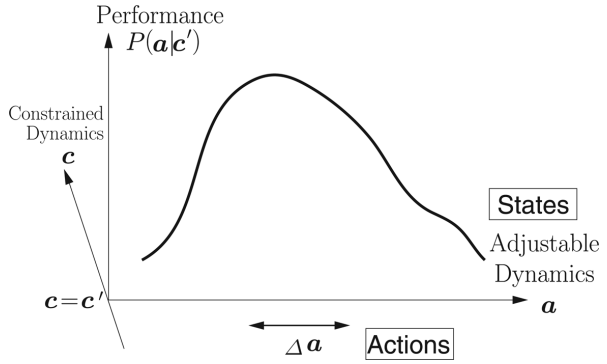


Fig. 2. Application of reinforcement learning to the dynamics matching problem.

dynamics matching. Consequently, the definition is arbitrary, and can be varied according to the problem.

Based on the above assumptions, the acquisition algorithm is proposed as follows.

(1) The adjustable dynamics $\boldsymbol{a}$ currently selected by the subject is recognized (state observation).

(2) The value function $V(\boldsymbol{a}, \Delta\boldsymbol{a})$ is examined, and the change $\Delta\boldsymbol{a}$ of the adjustable dynamics $\boldsymbol{a}$ is determined (action selection).

(3) Based on the change $\Delta\boldsymbol{a}$ determined in step (2), the adjustable dynamics $\boldsymbol{a}$ changes (state transition).

(4) The new adjustable dynamics $\boldsymbol{a}'$ is recognized (observation of state after transition).

(5) A task is performed under the new adjustable dynamics, and a reward is acquired (reward acquisition).

(6) Based on the goodness or badness of the reward, the value function V is evaluated and updated (update of value function)

(7) Return to step (1).

3. Target-Tracking Task

To investigate the usefulness of the above acquisition algorithm for dynamics matching, this section considers a target-tracking task using active vision as an example. In the system design it is not easy to achieve a balance between specificity and generality; the general discussion would become difficult in a sample problem based on a real machine, being affected by problems inherent to the real machine. In this paper, a simplified virtual system is considered as an example, and the method is evaluated from a general viewpoint.

3.1. Active vision and dynamics matching

Active vision is a vision system equipped with a device to move itself. In recent years, visual recognition has been considered more important in various problems, and better processing performance is required in active vision. In order to deal with the requirement, performance has been partially improved by hardware, such as speed-up of visual information processing [17]. Still, due to a large number of constraints, such as the speed and torque of the actuator, or the resolution and the size of the sensor, it is difficult to improve the performance by a simple algorithm. On the other hand, there is a severe constraint on the number of instruction steps in the embedded algorithm and the information processing strategy from the viewpoint of the real-time operation [18]. Thus, there arises the problem of matching the information processing dynamics to the hardware constraints and the real-time constraints.

This section discusses the problem in terms of the target-tracking task using active vision. The target-tracking task is the task of tracking a moving target using active vision. Figure 3 outlines the task. In order to simplify the discussion of the usefulness of the proposed method, a simplified virtual system is assumed. For simplicity, a point moving on a two-dimensional plane is defined as the target, and the camera is assumed to move on a two-dimensional plane which is parallel to the above. Time delay and observational error are ignored.

The system constructs an internal model of the target motion based on the observed information, and predicts the target trajectory up to several steps ahead. The trajectory is planned on the basis of prediction so that the hardware constraints are compensated, and the motion is actually performed. This processing series of "observation → model update → prediction → trajectory planning → motion" is iterated at a constant time interval.

3.2. Dynamics matching in target-tracking task

This section considers dynamics matching in the above task. The constrained dynamics $\boldsymbol{c}$ considered in Section 2.1 consists of the motion dynamics of the target, the limits of the camera and motor performance, and the computational load of the information processing. The adjustable dynamics $\boldsymbol{a}$ is the control of the information processing strategy.

In this investigation the constrained dynamics $\boldsymbol{c}$, the adjustable dynamics $\boldsymbol{a}$, and the performance P are defined as in Table 1. The number of look-ahead steps k is selected as the adjustable dynamics $\boldsymbol{a}$, which indicates how many steps ahead the motion should be predicted. It is an index expressing the computational complexity needed for prediction, and controls the strategy of how the computational

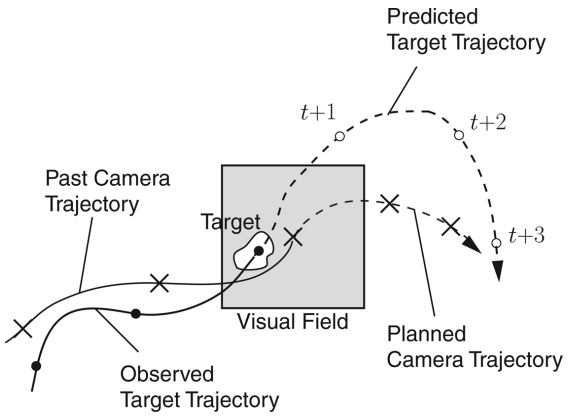


Fig. 3. Outline of a target-tracking task.

Table 1. Correspondence between the target-tracking task and dynamics matching problem

Constrained dynamics $\boldsymbol{c}$	Target dynamics c
	Width of camera view field d (mm)
	Maximum speed v of camera (mm/step)
	Upper limit of computation time $T = T_\mathrm{p} + T_\mathrm{t}$ (≤ 1) (step)
Adjustable dynamics $\boldsymbol{a}$	Number of look-ahead steps k
Performance P	Inverse of mean-square target prediction error 1/avg $(\|\boldsymbol{x}\|(t) - \boldsymbol{x}_\mathrm{p}(t)\|^2)$ (mm^{-2})

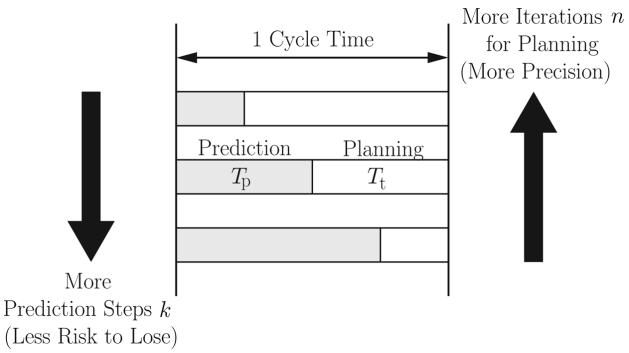


Fig. 4. Trade-off between the target position prediction and the camera trajectory planning.

complexity should be allocated between the target trajectory prediction and the camera trajectory planning.

In target trajectory prediction, if the look-ahead extends to the far future, the result can be utilized in the trajectory planning, so that severe constraints can be compensated. Thus, it is to be expected that the danger of losing the target will be decreased and the performance of the whole system will be assured. When k is increased to predict the far future, however, the processing time T_p will be increased. Moreover, in camera trajectory planning, when we wish to improve the accuracy of the trajectory, a longer time T_t will generally be required for iterative calculation and other processes. In order to maintain real-time operation, on the other hand, the processing time $T_\mathrm{p} + T_\mathrm{t}$ that can be spent in all processing is limited. If too much computational complexity is allocated to the target prediction, aiming at performance improvement, the time that can be allocated to trajectory planning is decreased. Then, as a result, the accuracy of the trajectory will be degraded and the target may be lost.

Thus, as shown in Fig. 4, there is a trade-off relation due to the contradiction between the increase in the amount of processing and the conservation of real-time operation. This trade-off situation is not known beforehand, and changes depending on the value of the constrained dynamics. Thus, it is necessary to control dynamically the balance of the computational complexity to be allocated to target prediction and trajectory planning. This compromise is implemented in this study by the control of the number of look-ahead steps k.

Thus, the problem reduces to the determination of the number of look-ahead steps k that minimizes the mean-square target prediction error under the constrained dynamics $\boldsymbol{c}$, that is, the determination of k^* such that

$$k^* = \operatorname*{argmax}_k P(k|c, d, v, T) \tag{1}$$

The system has no supervisor to suggest the optimal number of look-ahead steps k. Consequently, solution search by trial and error using reinforcement learning will be effective. In particular, in the on-line search for k, k must be varied in each trial and the target prediction error must be actually determined. In the next section, the algorithm proposed in Section 2.2 is applied to the above problem.

In order to improve performance in a real system, it will be necessary to consider more than one adjustable dynamical parameter at the same time, not a single parameter as in this study. However, since the method proposed in this paper uses reinforcement learning, the same algorithm can be used even if there are multiple parameters. For simplicity and for clear evaluation of the effectiveness of the algorithm, it is assumed that k is the only adjustable dynamics in the experiment. The case of multiple parameters is considered in Section 3.7.

3.3. Proposed parameter

Using the parameter proposed in Section 2.2, the formulation in the previous section is rewritten in the framework of reinforcement learning as in Table 2. This problem is equivalent to the search problem in a one-dimensional gridworld, where the numbers of look-ahead steps are arranged as a discrete array.

As the value function construction algorithm, we use Q-learning [16], which is a typical form of TD learning. The value function Q in Q-learning has the constrained dynamics $\boldsymbol{c}$ as the parameter. The update rule is expressed as follows:

$$\Delta Q(s_t, a_t \,|\, \boldsymbol{c}) = \alpha\Big(r_{t+1} + \gamma \max_a Q(s_{t+1}, a \,|\, \boldsymbol{c}) - Q(s_t, a_t \,|\, \boldsymbol{c})\Big) \tag{2}$$

Table 2. Description of variables in reinforcement learning

State s_t	Number of look-ahead steps k used by the system at time t
Action a	Increment number of look-ahead steps k by 1
	Decrement number of look-ahead steps k by 1
	Maintain number of look-ahead steps k
Reward r_t	100 × inverse of (mean-square prediction error in N steps + 1)

In the derivation of the action selection probability Pr (a), the softmax method [16]

$$\Pr(a) = \frac{\exp(\beta Q(s_t, a \mid \boldsymbol{c}))}{\sum_{a'} \exp(\beta Q(s_t, a' \mid \boldsymbol{c}))} \tag{3}$$

is used. Figure 5 shows the block diagram of the algorithm.

3.4. Experimental formulation

Based on the above algorithm, a numerical experiment was performed for the target-tracking task. The experiment was set up as follows.

3.4.1. Constrained dynamics

As the movement of target c, three different ellipses with major axes from 200 mm to 400 mm and with the minor axis constant at 100 mm, together with Brownian motion based on a pseudo-random number, were used. As regards the camera hardware, the edge of the view field, d, was varied over six values from 250 mm to 500 mm. The maximum speed of movement, v, was set to nine values from 100 mm/step to 300 mm/step. These constrained dynamics variables were switched at random for each trial. The constrained dynamics T for the processing time was set constant, with a value of 1, in this study.

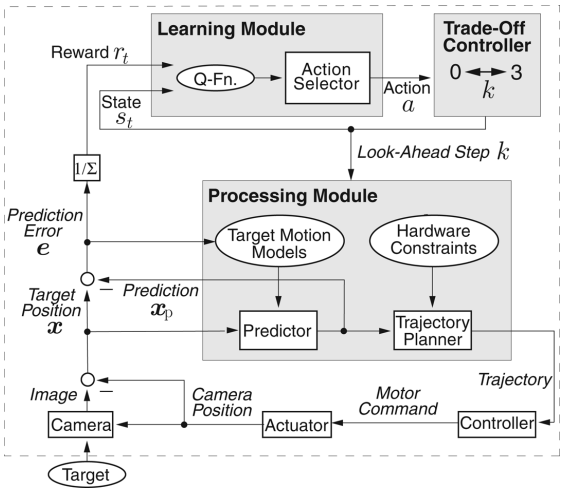


Fig. 5. Structure of the proposed algorithm.

3.4.2. Adjustable dynamics

It was assumed that the number of look-ahead steps k had one of four values, 0, 1, 2, and 3. $k = 0$ implies that prediction was not applied, and the present position of the target was defined as the goal. In this case, the tracking error was regarded as the target prediction error.

3.4.3. Target prediction

In order to track the target by feedforward control, it is necessary first to construct an internal model of the target motion $\boldsymbol{x}(t)$. It was assumed that an internal model for each kind of target c was retained individually, and that the two-dimensional position $\boldsymbol{x}(t)$ of the target was represented as $\boldsymbol{x}(t+1) = A\boldsymbol{x}(t) + \boldsymbol{B}$, using the two-dimensional state-space representation $\{A, \boldsymbol{B}\}$ for each kind. It was assumed that the internal model itself was not clearly known at the start of learning (white noise was superimposed on the true value). The estimate was successively updated in parallel to the tracking, using the method of steepest descent based on the prediction error. The prediction $\boldsymbol{x}_p(t+1)$ was calculated from this model. When the number of look-ahead steps was k, $\boldsymbol{x}_p(t+k)$ was determined by iteratively applying the procedure.

3.4.4. Trajectory planning

Since the speed of movement of the camera is limited, it may happen that tracking along the predicted target trajectory cannot be performed. Consequently, the camera trajectory should be planned carefully. Optimal control is generally used in trajectory planning for robots, where approximate calculations are iterated since the system is nonlinear. The following trajectory planning algorithm is an abstraction of the iterative approximate calculation used in conventional methods.

In trajectory planning in this experiment, accuracy of the camera trajectory was successively improved, as shown in Fig. 6. The initial setting of the camera trajectory was defined as the straight line trajectory $\boldsymbol{X}(t+i)$ ($i = 1, \ldots, k$) toward the point $\boldsymbol{x}_p(t+k)$. Within the range of the maximum speed of movement v, this straight-line trajectory was iteratively modified to approach the target trajectory, using the method of steepest descent:

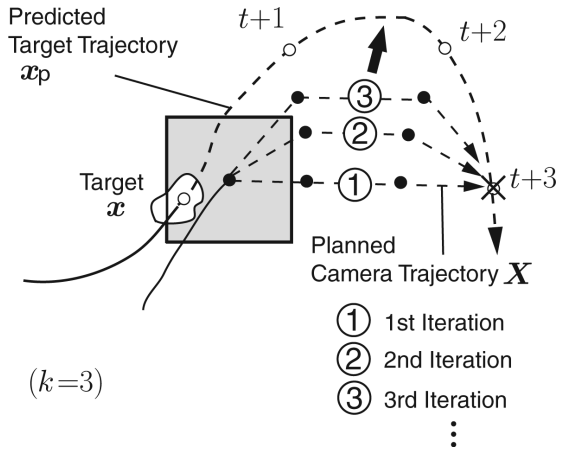


Fig. 6. Camera trajectory planning.

$$\Delta \boldsymbol{X}(t+i) = \mu(\boldsymbol{x}_{\mathrm{p}}(t+i) - \boldsymbol{X}(t+i)) \quad \text{for } i = 1, \ldots, k$$

Here μ is a positive constant, and should be small in order to assure stability.

Based on the concept of dynamics matching, the computation time for the target prediction and the trajectory planning should be explicitly considered. The processing time was limited to a constant *T*. Consequently, when the look-ahead *k* is increased, the processing time for target prediction increases, and the time that can be allocated to trajectory planning decreases. For simplicity, this constraint was implemented by defining the penalty that the number of iterative calculations *n* for trajectory planning was decreased by νk (where ν is some positive constant), that is, in proportion to *k*.

When the target was lost en route, the operation shifted to the search mode and the target was sought by moving the camera at random with a constant speed. In the process, target prediction and trajectory planning were not applied. When the target was captured in the view field, the operation escaped from the search mode and tracking was restarted.

3.4.5. Learning parameters

Various parameters such as those in Q-learning were set as follows: $\alpha = 0.01$, $\beta = 0.05 \times$ (experienced number of episodes), $\gamma = 0.999$, $\mu = 0.1$, $n = 9$, $\nu = 2$, $N = 30$.

3.5. Experimental results

3.5.1. Relation between severity of constraints and number of look-ahead steps

Figure 7(a) shows the distribution of the look-ahead steps averaged in the last 2000 episodes, when the view

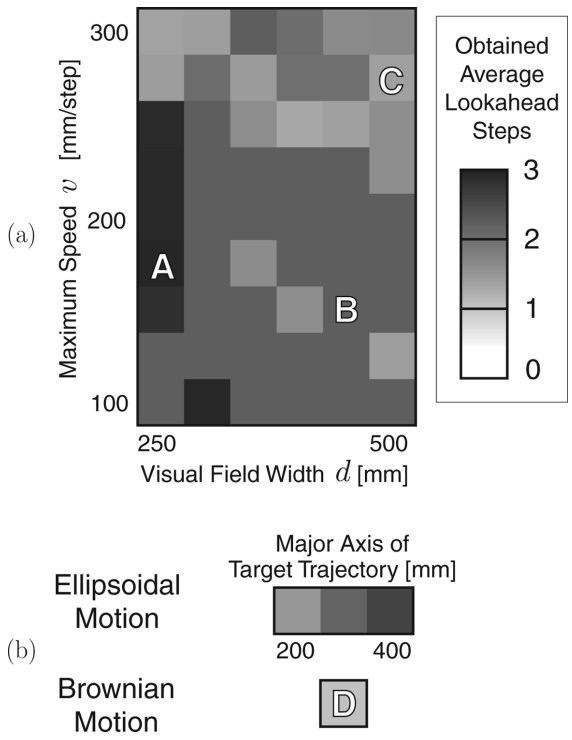


Fig. 7. Distribution map of averages of obtained look-ahead steps: (a) under sensory-motor constraints, (b) under external constraints.

angle *d* and the maximum speed *v* were varied independently. The darkness of each block represents the number of look-ahead steps for each constrained dynamic. Similarly, Fig. 7(b) is the average number of look-ahead steps when target trajectory *c* is varied (the horizontal axis in the upper chart represents the major axis of the ellipse).

It is evident from Fig. 7 that the number of look-ahead steps tended to increase when the constraints were severe (i.e., the view field was narrow, the maximum speed of movement of the camera was low, and the target speed was high). This can be interpreted as follows.

When constraints are not severe, the risk that the target will be lost at the next instant is minimized so long as the target is set at the center of the view field. This can be achieved in this trajectory planning framework only by setting a single look-ahead step in target prediction and trajectory planning. If further target prediction and trajectory planning were performed, there would be a danger that the performance might be degraded due to the computational load penalty. It is considered that the system "found" such a situation by trial and error by reinforcement learning, and reduced the number of look-ahead steps to a low value.

When the constraints are severe, on the other hand, the target may easily be lost if such a strategy is used. Then, the system took the strategy of compensating the con-

straints by increasing the look-ahead steps in target prediction and trajectory planning, and assuring performance up to several instants ahead, even though the computational load would be increased. When the target showed Brownian motion, the system "found" that look-ahead had no significance and reduced the number of look-ahead steps.

In Fig. 7, the distribution of the obtained look-ahead steps is not necessarily uniform. The reason may be that the performance was affected by slight changes of the parameters in such a severe environment as in this experiment. The improvement of this aspect is left for future study.

3.5.2. Convergence of number of look-ahead steps

Figure 8 shows the change of the state (number of look-ahead steps) at each of the points A, B, C, and D in Fig. 7. It is evident that the number of look-ahead steps converged to a constant at each point. At point D, a longer time was required for convergence than at the other three points. In addition, the value itself was unstable. The reason seems to be that the target motion differed each time, and the optimal number of look-ahead steps would vary between 0 and 1.

3.5.3. Performance improvement

Figure 9 compares the behavior of the value function Q for the case in which k was adaptively varied (proposed method) and for the cases in which k was fixed as 0, 1, 2, or 3 (only the action of retaining each value of k was performed). We set d = 450 mm, v = 150 mm/step. For the proposed method, the value function for the action to retain k = 2 is shown. It is evidently optimal to set k = 2. In the proposed method, the system "found" that it is optimal, and finally succeeded in converging to the value function with k = 2 fixed. Since the value function greatly reflects the performance, it is concluded that the proposed method was useful in acquiring a high-performance information processing strategy.

Figure 10 shows the behavior of the root-mean-square prediction error at point B in Fig. 7. It can be seen that the error decreased (although there seems to be an upper bound, due to the properties of the numerical experiment; such trials correspond to the situation in which the target was lost in the early stage and a time-out occurred). It is also verified that the error decreased similarly for the other points. Thus, it is concluded that the proposed method actually improved the performance.

3.5.4. Difference of information processing strategy observed in behavior

The difference of the information processing strategy was also reflected in the macroscopic behavior. Figure 11 shows typical cases of tracking that occurred in the case of severe physical constraints (d = 300 mm, v = 250 mm) and in the case of nonsevere constraints (d = 300 mm, v = 400 mm). When the constraints were not severe, the number of look-ahead steps was set as 1 and the target was captured at the center of the view field. Thus, tracking almost exactly along the target trajectory was achieved. When the constraints were severe, on the other hand, a reasonable strategy was acquired as follows. The number of look-ahead steps was set as 2, so that the periphery of the screen was effectively utilized and the target was tracked without moving the camera much. This is a tracking strategy suited to active vision, when the actuator is severely constrained.

3.6. Adaptability to environmental variations

In order to investigate the usefulness of the adaptive method by on-line learning, an experiment was performed in a varying environment. Specifically, the target trajectory, which is one of the constrained dynamics, was changed suddenly in the experiment, and it was examined whether or not the system could switch to the optimal number of look-ahead steps. It was assumed that the system can detect the change of the target trajectory (the detection procedure is not discussed in this paper, but various methods have been proposed [19–24]), and reset the inverse temperature parameter β in the softmax method to the initial value (it should be noted that the procedure aims to restore the

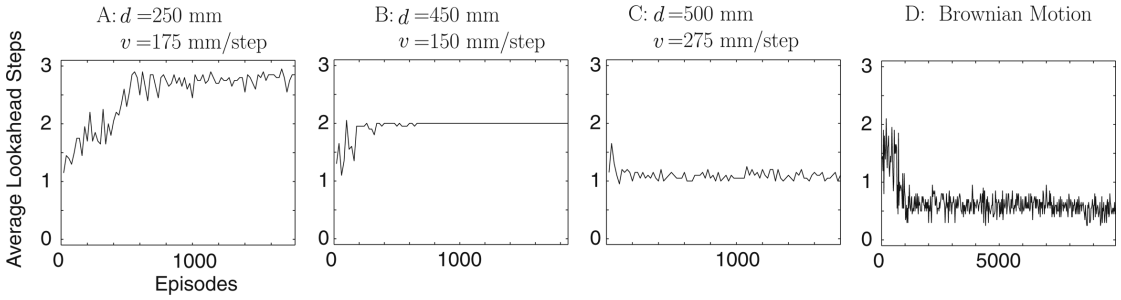


Fig. 8. Changes in look-ahead steps (average values per 20 episodes).

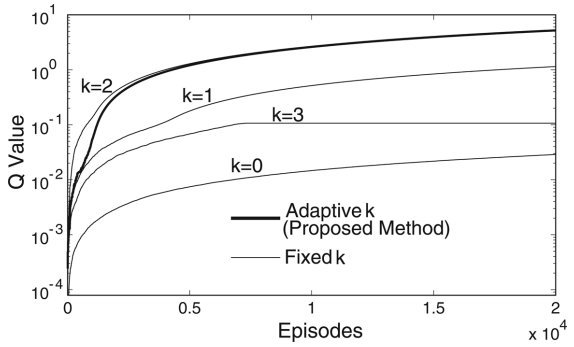


Fig. 9. Comparison of performance with adaptive k (proposed method) and fixed ks.

learning ability, which has degraded exponentially with time, and not to restore the result of learning). In the proposed method, the content of the previous learning was overwritten by the new content. It may technically be possible to retain the previous content, but this topic is not discussed in this paper.

Figure 12 shows the change in the number of look-ahead steps, in the case that the major axis of the elliptic target trajectory was changed from 350 mm to 250 mm, after 5000 episodes elapsed. It can be seen that the number of look-ahead steps was 3 before the change of the trajectory, but was switched to 2 after the change. Considering that the constraint was severer before the trajectory change, we see that the system achieved dynamics matching according to the varying environment.

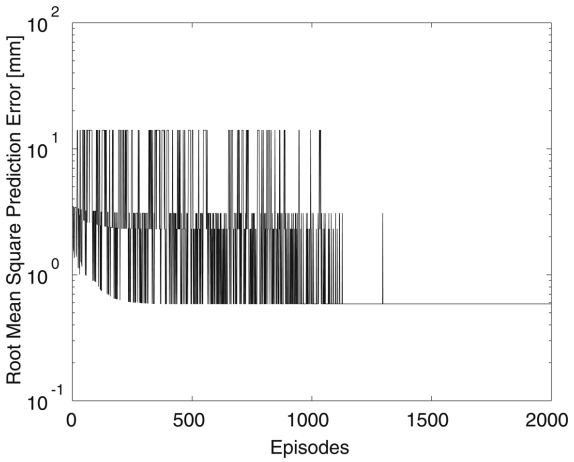
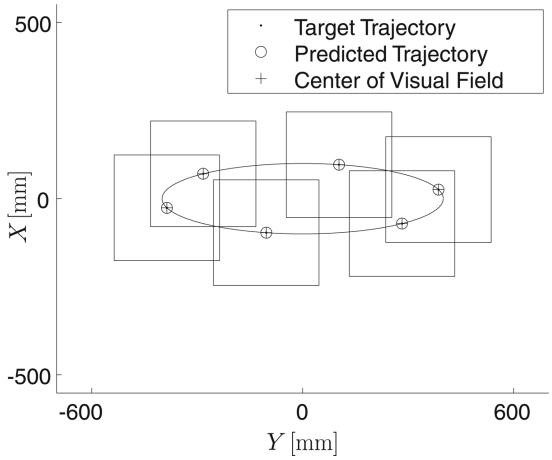


Fig. 10. An example of changes in root mean square prediction error.



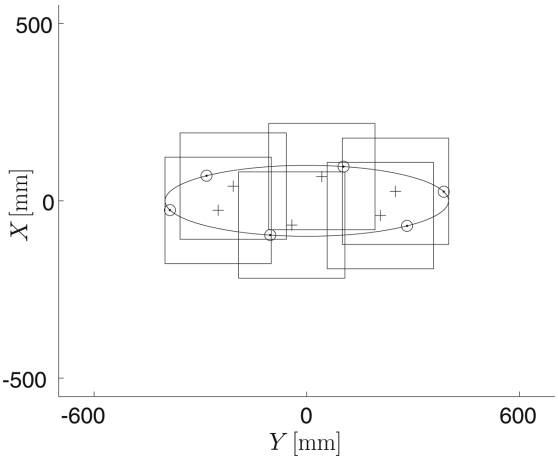


Fig. 11. Behavior of tracking. Upper: under loose constraints (1-step look-ahead); lower: under severe constraints (2-step look-ahead).

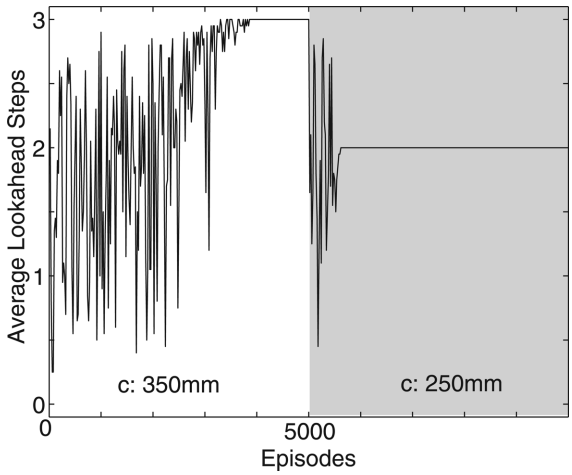


Fig. 12. Changes in look-ahead steps according to the shift in the target trajectory.

3.7. Applicability to more complex problems

The experiment up to this stage has been set equivalent to the one-dimensional gridworld problem in order to demonstrate clearly the effectiveness of the algorithm. However, the dynamics matching problem in a real system is often of higher dimension and more complex. The algorithm presented in this paper is directly applicable, in principle, to higher-dimensional problems. It should be noted, however, that the learning time tends to increase with the dimension. Below, the applicability to more complex problems is discussed from the viewpoint of the learning time.

The learning process includes a large number of stochastic elements, and strict evaluation of the learning time is essentially impossible. However, it is possible to provide a rough estimate of the learning time. Whitehead has shown that the learning time in Q-learning increases exponentially with the size of the state space [25].

The elements which have been considered in the experiment up to the previous section are two: the processing time T_p for prediction of the target trajectory, and the processing time T_t for camera trajectory planning. A further element is added, namely, the processing time T_s in supervised learning of the internal model of the target. Thus, the task consisting of three elements is considered. Specifically, it is as follows.

The number of iterations of the method of steepest descent for acquiring the internal model of the target was defined as another adjustable dynamics k', and a penalty was defined in which the number of iterations in the trajectory planning was reduced according to the value of k'. The problem then is equivalent to the two-dimensional gridworld problem with k and k' as the parameters. The value function is four-dimensional, being increased by two. In the experiment, the number of iterations k' of the method of steepest descent was set to two values, 1 and 15, or to four values, 1, 5, 10, and 15. Other parameters were fixed as v = 150 mm/step, d = 450 mm, c = 300 mm. The learning was judged to be completed when the selected k and k' both remained constant for 200 episodes.

Table 3 shows the number of episodes until learning was completed (the average over 10 trials). It is verified that the learning time actually increased when the problem became more complex. When the number of elements in the value function was large, being 144, there existed a few cases in which learning did not converge even if 20,000 episodes were performed (these cases were excluded from the statistics in Table 3). When the number of elements was 72, a strategy in which the priority was given to tracking rather than model learning was clearly acquired. When the number of elements was 144, however, there was no clear pattern in the acquired strategy, suggesting the presence of a complicated trade-off relation among the three.

Table 3. The number of episodes spent for learning

Number of adjustable dynamics	Number of value function elements	Number of episodes
1 only k	$4 \times 3 = 12$	1432
2 k and k'	$(4 \times 3) \times (2 \times 3) = 72$	2616
2 k and k'	$(4 \times 3) \times (4 \times 3) = 144$	4148

The expression "$(4 \times 3) \times (2 \times 3)$" means that k has four states and three actions, and k' has two states and three actions.

In general, the real machine does not have so many degrees of freedom and adjustable dynamics. Consequently, it is expected that cases where solution is impossible will not arise very often. When the number of degrees of freedom is high, however, the learning time will increase explosively, limiting the application of the proposed method. Consequently, in real design it will be necessary to estimate the learning time and to determine the number of degrees of freedom to be acquired. To reduce the learning time, it will be effective to utilize various high-speed algorithms [25, 26].

Furthermore, a larger amount of resources must be allocated to the value function itself when the dimension is increased. However, this will not be a serious problem in dynamics matching, considering recent advances in large-capacity memory. A unified discussion of the resources to be allocated to the value function is difficult, since it depends greatly on the task. However, many techniques have been proposed to reduce the resources used for the value function by compression and feature parameter representation [27, 28], and it will be effective to apply these techniques whenever necessary.

It is also noted that, while the inverse of the mean-square prediction error was used as the performance measure in the experimental results presented in this paper, the results were almost identical when the inverse of the mean-square tracking error was used.

These results clearly show that the proposed method is useful. It is indispensable to achieve dynamics matching in order to improve the speed of sensory-motor fusion systems such as robots. The trade-off relation as verified in this paper is essential and independent of tasks. Consequently, the proposed method will be a versatile and effective means of solving the problem.

4. Conclusions

This paper has modeled dynamics matching as an optimization problem, and has constructed an adaptive

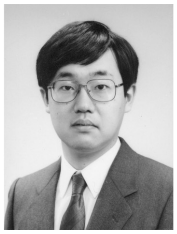
acquisition algorithm based on reinforcement learning. The method was applied to a target-tracking task using active vision. It was shown that the proposed method is useful in realizing dynamics matching by acquiring a reasonable tracking strategy according to the situation.

Problems left for the future include improving the convergence of the algorithm, and demonstrating the effectiveness of the method by implementation in a real active vision system [17].

REFERENCES

1. Ishikawa M, Ogawa K, Komuro K, Ishii I. A CMOS vision chip with SIMD processing element array for 1 ms image processing. Dig Tech Papers of 1999 IEEE Int Solid-State Circuit Conf (ISSCC'99), p 206–207.
2. Kaneko M, Tsuji T, Ishikawa M. The robot that can capture a moving object in a blink. Proc 2002 IEEE Int Conf Robotics and Automation (ICRA'02), p 3643–3648.
3. Guertin JA, Townsend WT. Teleoperator slave—WAM design methodology. Industrial Robot 1999;26:167–177.
4. Robotics Society of Japan (editor). Robotics handbook. Corona Publishing Co.; 1990.
5. Namiki A, Nakabo Y, Ishii I, Ishikawa M. 1-ms sensory-motor fusion system. IEEE/ASME Trans Mech 2000;5:244–252.
6. Namiki A, Komuro T, Ishikawa M. High-speed sensory-motor fusion based on dynamics matching. Proc IEEE 2002;90:1178–1187.
7. Mason MT, Salisbury JK. Robot hands and the mechanics of manipulation. MIT Press; 1985.
8. Spong MW, Vidyasagar M. Robot dynamics and control. John Wiley & Sons; 1989.
9. Hollerbach JM. Dynamic scaling of manipulator trajectories. In: Richards W (editor). Natural computation. MIT Press; 1988. p 455–464.
10. Sawaragi T. Temporal and environmental contexts and uncertainties in modeling. J Robotics Soc Japan 2000;18:318–324.
11. Beach LR, Mitchell TR. A contingency model for the selection of decision strategies. Acad Manage Rev 1978;3:439–449.
12. Nigam R, Lee CSG. A multiprocessor-based controller for the control of mechanical manipulators. IEEE J Robot Autom 1985;1:173–182.
13. Butazzo GC. Hard real-time computing systems—Predictable scheduling algorithm and applications. Kluwer Academic; 1997.
14. Singh S, Bertsekas D. Reinforcement learning for dynamic channel allocation in cellular telephone systems. Advances in Neural Information Processing Systems 9 (NIPS*96). MIT Press; 1997. p 974–980.
15. Hall LO, Pokorny MA. Averaged reward reinforcement learning applied to fuzzy rule tuning. Proc Int Conf Fuzzy Logic and Applications, 1997.
16. Sutton RS, Barto AG. Reinforcement learning: An introduction. MIT Press; 1998.
17. Nakabo Y, Ishikawa M, Toyoda H, Mizuno S. 1 ms column parallel vision system and its application of high speed target tracking. Proc 2000 IEEE Int Conf Robotics & Automation (ICRA2000), p 650–655.
18. Ishii I, Ishikawa M. High speed target tracking algorithm for 1 ms visual feedback system. J Robotics Soc Japan 1999;17:195–201.
19. Sakaguchi Y, Takano M. Learning to switch behaviors for different environments: A computational model for incremental modular learning. Proc 2001 Int Symp Nonlinear Theory and its Applications (NOLTA2001), p 383–386.
20. Wolpert DM, Kawato M. Multiple paired forward and inverse models for motor control. Neural Networks 1998;11:1317–1329.
21. Haruno M, Wolpert DM, Kawato M. MOSAIC model for sensorimotor learning and control. Neural Comput 2001;13:2201–2220.
22. Tani J, Nolfi S. Learning to perceive the world as articulated: An approach for hierarchical learning in sensory-motor systems. Neural Networks 1999;12:1131–1141.
23. Ishii S, Yoshida W, Yoshimoto J. Control of exploitation-exploration meta-parameter in reinforcement learning. Neural Networks 2002;15:665–687.
24. Mizuno J, Murakoshi K. A parameter control method inspired from neuromodulators in reinforcement learning. Tech Rep IEICE 2002;NC2002-102.
25. Whitehead SD. A complexity analysis of cooperative mechanisms in reinforcement learning. Proc 9th National Conf Artificial Intelligence (AAAI-91), Vol. 2, p 607–613.
26. Vijayakumar S, Schaal S. Fast and efficient incremental learning for high-dimensional movement systems. Proc 2000 IEEE Int Conf Robotics and Automation (ICRA2000), p 1894–1899.
27. Shewchuk J, Dean T. Towards learning time-varying functions with high input dimensionality. Proc 5th IEEE Int Symp Intelligent Control, p 383–388, 1990.
28. Sutton RS, Whitehead SD. Online learning with random representations. Proc 10th Int Conf Machine Learning, p 314–321, 1993.

AUTHORS (from left to right)

Naoko Ogawa received her B.E. degree from the Department of Mathematical Engineering and Information Physics of the University of Tokyo in 2000 and completed the M.E. program in mathematical engineering and information physics in 2002 and Ph.D. program in information science and technology in 2005. She has been engaged in research on sensor fusion, machine learning, and control of microorganisms. She is a member of IEEE, Virtual Reality Society of Japan, Robotics Society of Japan, and SICE.

Yutaka Sakaguchi received his B.E. degree from the Department of Mathematical Engineering and Information Physics of the University of Tokyo in 1986, completed the M.E. program in mathematical engineering and information physics in 1988, and became a research associate there. He has been an associate professor in the Graduate School of Information Systems, University of Electro-Communications, since 1994. He is principally engaged in research on sensory, perceptual, and motor control systems of the human brain. He is a member of the Japan Neural Network Society, Japan Neuroscience Society, Vision Society of Japan, Japanese Cognitive Science Society, Society for Neuroscience, Vision Sciences Society, and IEEE.

Akio Namiki received his B.E. degree from the Department of Mathematical Engineering and Information Physics of the University of Tokyo in 1994 and completed the M.E. program in mathematical engineering and information physics in 1996 and doctoral program in 1999. He became a JSPS research associate in 1996 and CREST researcher in 2000. He has been an assistant professor in the Department of Information Physics and Computing, Graduate School of Information Science and Technology, University of Tokyo, since 2004. His research interests are sensor fusion, intelligent robots, and multifinger hand control. He is a member of the Robotics Society of Japan and Japan Society of Mechanical Engineers.

Masatoshi Ishikawa (member) received his B.E. degree from the Department of Mathematical Engineering and Instrumentation Physics of the University of Tokyo in 1977, completed the M.E. program in 1979, and joined the Industrial Products Research Institute, Ministry of International Trade and Industry. He became an associate professor (1989) and professor (1999) in the Department of Mathematical Engineering and Information Physics of the University of Tokyo, and has been a professor in the Department of Information Physics and Computing, Graduate School of Information Science and Technology, since 2001. His research interests are vision chips, optical computing, and sensor fusion.