Multiple Self-Organizing Maps for Control of a Redundant Manipulator with Multiple Cameras

Nobuhiro Okada, Jinjun Qiu and Eiji Kondo
Kyushu University
Japan

1. Introduction

Vision guide for a manipulator has been one of the major research issues in robotics. Coordination schemes of visuo-motor systems can be classified on the basis of the knowledge about manipulator kinematics and camera parameters. Many researchers have proposed a number of systems that deal with unknown manipulator kinematics and unknown camera parameters. In the studies, visuo-motor models are either estimated analytically during the execution of tasks on-line or learned prior to the execution off-line. Artificial neural networks can be used to learn the non-linear relationships between features in images and the manipulator joint angles. Miller et al. proposed a neural network based on the learning control system, where a cerebeller model arithmetic computer memory was employed for the learning (Miller, 1989). Carusone et al. used a network to train an un-calibrated industrial robot (Carusone & Eleurterio, 1998). In their systems, neural networks provided the estimation of the poses of targets in the manipulator coordinate frames, and the poses were used to guide the manipulator to grasp the objects. However, supervisors were needed in the systems.
Self-organizing map (SOM) based on the Kohonen algorithm is an important unsupervised artificial neural network model (Kohonen, 1998). It has shown great potential in application fields such as motor control, pattern recognition, optimization, and so on, and also has provided insights into how mammalian brains are organized (Wiener et al., 2000) (Behera & Kirubanandan, 1999). During the past years it has been demonstrated that the SOM can solve the inverse kinematics problem for visuo-motor control. Buessler et al. determined arm movements by tracking an image target (Buessler & Urban, 1998) (Buessler et al., 1999). The correlation between an image-defined error and the joint movement was learned on-line using self-organizing algorithm for making the error zero. Multiple neural maps were combined to simplify neural learning in their study. Martinetz et al. and Walter et al. used a three dimensional lattice to learn the nonlinear transformation that specifies the joint angles of a 3-DOF manipulator so that the angles take the tip of the manipulator to a target point given in the coordinates provided by two cameras (Marinetz et al., 1990) (Walter & Schulten, 1993). In all of these studies, however, they solved the visuo-motor coordination problems

with non-redundant manipulators in an environment without obstacles. Such obstacle avoidance problems are important for manipulators that work in real environments. Zeller et al. developed a motion planning for a non-redundant manipulator to avoid collision with obstacles in a cluttered environment by using the TRN model (Zeller et al., 1997). They used a fact that a locally optimized path can be determined by minimizing the Euclidean distance from the current position to a given goal. Collision check was performed not in the self-organizing process but in the path planning process afterwards. Collobert developed a new organizing principle for perceptual systems based on multiple Kohonen self organizing maps. (Collobert, 2006) These maps are arranged in order to model the global brain activity as seen on tomography pictures. In contrast to these precedent studies, our system is not only for precise positioning of the end-effector but also for ensuring obstacle free poses of the manipulator using multiple SOMs. We intend to realize coordination for a visuo-motor system with a redundant manipulator in a cluttered environment. The redundancy is then used to make the manipulator take obstacle free poses and achieve high manipulability.

In the previous researches, Zha et al. used a SOM to coordinate a visuo-motor system in an environment with obstacles (Zha et al., 1996). Collisions between the links and obstacles were, however, not well considered. We introduced a potential field to avoid such collisions only in a 2D space (Okada et al., 1999). Han et al. realized collision avoidance for a visuo-motor system in a 3D space (Han et al., 2003). The occlusion problem was, however, not solved effectively even in the system.

Vision systems are generally classified by the number of cameras, camera configurations, the level of calibration and some a priori knowledge about the scene. The binocular configuration is a commonly used configuration. In comparison with the eye-in-hand configuration, it allows a wide field of view and then it makes easy to observe both the manipulator and targets simultaneously. Such a vision system was employed in Han et al.'s work. However, since they treated spaces occluded by obstacles in the image space as unreachable spaces for the manipulator, the workspace was restricted.

In order to handle the occlusion problem, we have developed a visuo-motor system with multiple related SOMs and a redundant camera system in this paper. The SOMs are directly connected to the cameras and learn to perform manipulator control. Based on the visibility of a target given in the workspace, the appropriate map is selected. The map outputs a joint angle vector which makes the manipulator reach the target with an obstacle free pose. The proposed learning algorithm ensures that the manipulator moves smoothly and consistently in the whole workspace no matter which map is selected. The advantages of the proposed method are: (1) By employing multiple maps, the system overcomes the occlusion problems in cluttered environments. The cooperation and complementation of maps make the manipulator consistently move in the whole workspace. (2) In our self-organizing learning procedure, the visuo-motor system learns not only to position the end-effector precisely but also ensure that the manipulator takes obstacle free poses.

2. Our Visuo-Motor System

Our visuo-motor system is illustrated in Fig.1. The system contains a 4-DOF redundant manipulator, multiple CCD cameras, and multiple related SOMs. The CCD cameras are used to get the target positions, the locations of the end-effector and the manipulator poses. They also acquire information about obstacles by using simple image technique. From visual

information provided by the cameras, the SOMs learn projections that convert the position vectors of the targets in the image spaces into the joint angle vectors of the manipulator.
Although stereo camera systems can provide 3D information and we have used such a system in our previous works, the system could not well deal with spaces occluded by obstacles. They introduced 3-cameras system to overcome the situation (Han et al., 2006). However, the result was limited. To deal with the occlusion problem, a multiple camera system is presented in this paper. The valid workspace is extended by using the cameras at multiple viewpoints. Related SOMs are simultaneously employed in the visuo-motor system.
Assume that the projections of a target point in the camera images are $u_{t\text{-}S1}$, $u_{t\text{-}S2}$, $u_{t\text{-}S3}$ (side camera1, 2, 3) and $u_{t\text{-}T}$ (top camera). A pair of image coordinates of the top camera and another side camera is combined into a 4-dimensional vector $u_{t\text{-}SOMn}$, and then it is used as an input to one of SOMs. Since the valid workspaces of the maps are different from each other, the maps are alternately used.

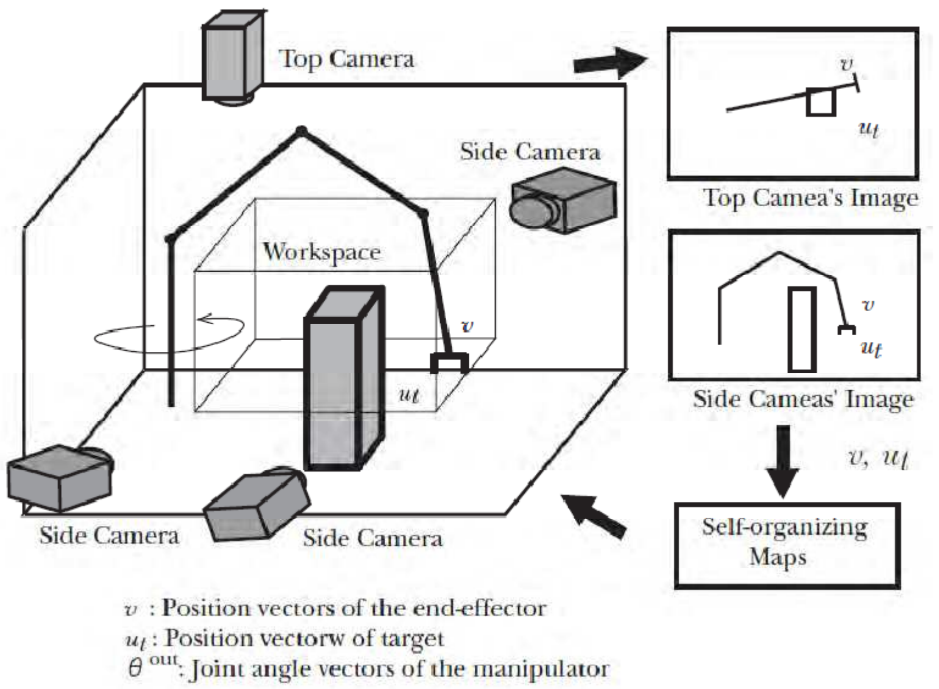
Top Camera
Side Camera
Workspace
v
u_t
Side Camera
Side Camera
Top Camea's Image
Side Cameas' Image
v, u_t
Self-organizing Maps
v : Position vectors of the end-effector
u_t : Position vectorw of target
θ^{out}: Joint angle vectors of the manipulator

Fig. 1. Outline of our visuo-motor system

As shown in Fig.2, each SOM consists of neurons, which are distributed in 2 image spaces that correspond to cameras used as inputs. Each neuron has the following 4 parameters.
1. W: Position of the neuron in 2 image spaces.
2. J: Jacobi matrix from the manipulator joint angle space to the image spaces.
3. θ: Joint angle at W.
4. ξ: Gradient vector of the evaluation function H.
When a target u_t is given in the workspace, one of the maps is selected based on which camera can see the target. In the selected map, then, the neuron nearest to the projection of the target is chosen. The manipulator joint angle vector is finally calculated obeying the linear function (1).

$$\boldsymbol{\theta}_{out} = \frac{\sum g_n \left(\boldsymbol{\theta} + \mathbf{J}^{\dagger}\left(\mathbf{u}_t - \mathbf{W}\right)\right)}{\sum g_n} \tag{1}$$

Here J† is a pseudo-inverse matrix of J. Even though the projection from the image spaces to the joint angle space is not linear for a PUMA type manipulator, such a linear approximation can still be used in a small areas. In our SOMs, since many neurons are distributed in the image spaces, the area controlled by a neuron will be small enough.

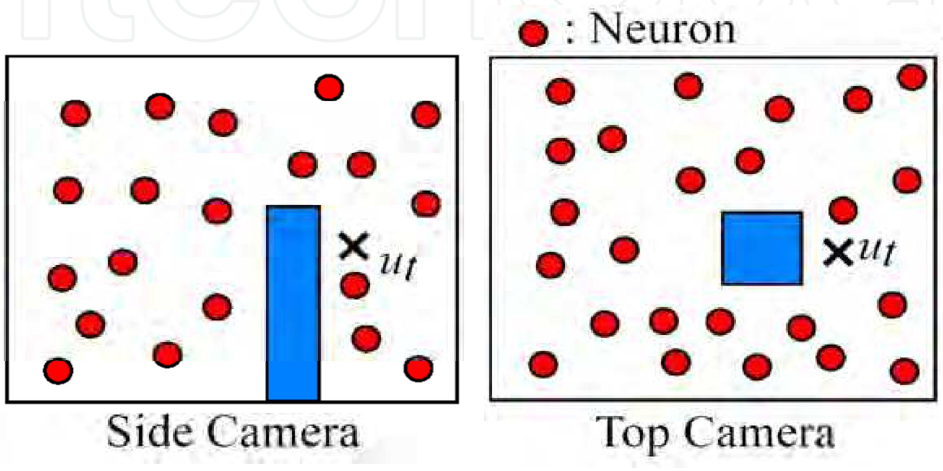

Fig. 2. A self-organizing map used in our system

In our actual system, weighted sum of outputs from multiple neurons around the target is used instead of (1). Where g_n is the weight defined by the following equation.

$$g_{nt} = \begin{cases} \exp(-n/\lambda) & \text{for } \exp(-n/\lambda) > \varepsilon \\ 0 & \text{for } \exp(-n/\lambda) \leq \varepsilon \end{cases} \tag{2}$$

In the equation, n is the order of the neuron determined according to the distance between the neuron and the target. Nearer the target it has a larger value and farther it has a smaller value. The symbols λ and ε are values for defining the number of neurons that can affect θ_{out}.

3. Learning Procedure Of The Self-Organizing Maps

3.1 Reason for multiple SOMs

In our system, multiple SOMs are employed. If we utilize one SOM to corresponding to multiple cameras, the SOM will have a high dimensional space and be computationally difficultly. On the other hand, the distance between neurons in the SOM will be too huge to use a linear approximation. In addition, when a camera cannot see a target, the input of the SOM will become incomplete, therefore, the neurons cannot be all updated. To solve these problems, we utilize multiple SOMs.

If these multiple SOMs learn separately, therefore, outputs from them will be different with each other even for the same target in the workspace. This will result in that when the

system switches the outputs used for the manipulator control, the manipulator moves inconsistently. The problem should be effectively removed in the learning process. Then the problem can be described as: the learning algorithm has to guarantee that the manipulator moves smoothly and consistently in the whole workspace no matter which SOM is used for control.
Our learning procedure is explained using Fig.3. In the learning process, a target position u_t is randomly given in the workspace and the cameras obtain the position. At first we assume that the top camera (Fig.1) can always see the target in our former system. Each SOM is respectively related to a side camera and it receives target information from the side camera and the top camera (therefore the information is a 4-dimensional vector). Depending upon which side cameras can see the target, corresponding maps update their parameters. When only one side camera can see the target the corresponding SOM updates its parameter by itself. When more than 2 cameras can see it, the corresponding SOMs learn under influences from others. In the case, one of SOMs is arbitrary chosen to output the joint angles θ_{out}, and the manipulator is driven by it. Then all cameras that can see the end-effector obtain its position v. Finally each SOM related to the camera updates its parameters using u_t, θ_{out} and v. To remove the assumption that the top camera can always see the target, we increased the number of SOMs. The outputs from every 2 cameras form one SOM. The camera which can see the target is treated as the top camera of our former system. The SOMs learn using the learning algorithm motioned above. If more than 1 camera can see the target, the corresponding SOMs learn simultaneously. The updating algorithms will be described in the following subsections. The learning is done by iterating the above process for many targets.

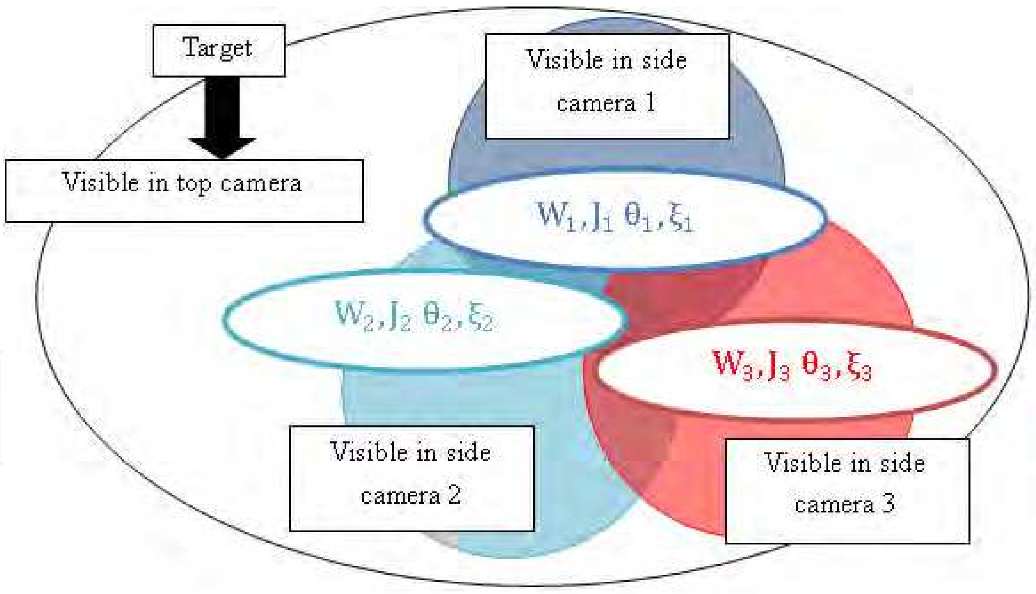

Fig. 3. Relation of multiple SOMs in the learning process

This learning procedure results in that the neurons of the multiple SOMs for a target possess similar values of θ_{out} in the end of learning even though other parameters may differ from each other. Thus the outputs from any SOMs will ensure the manipulator takes the same pose. By one of the major features of SOM, an assignment of similar joint angles to adjacent targets is also achieved. Both features bring out a continuous and smooth transformation from the input image spaces of target positions to the output spaces of joint angles. Hence the SOMs guarantee smooth and consistent movements of the manipulator in the whole workspace.

3.2 SOM's learning procedure

A. Evaluation function

Calculating the inverse kinematics for a given end-effector position is a hard problem for redundant manipulators since it is an ill-posed problem that has many possible solutions. We introduced evaluation functions to resolve the under-determination into the system. In addition to making the end-effector reach the target position, the system outputs the joint angle configuration that optimizes the functions. One evaluation function is to achieve high manipulability, and the other is to make the manipulator take an obstacle free pose. The functions are respectively shown in (3) and (4). The total evaluation function is defined as the weighted sum of H_M and H_O (5). The function H_M is for manipulator with high manipulability, the other function H_O for manipulators with free obstacle poses.

$$H_M = \sqrt{\det\left(\mathbf{J}(\boldsymbol{\theta})\mathbf{J}^T(\boldsymbol{\theta})\right)} \tag{3}$$

$$H_O = \sum_l \left(1.0 - \frac{D_0}{d + D_0}\right) \tag{4}$$

$$H = \alpha_1 H_M + \alpha_2 H_O \tag{5}$$

Here d is the shortest distance from each link to the obstacles, and D_0 is the predefined value that provides the effective area of the potential. α_1 and α_2 are weights which are determined depending on the desirability of the individual functions.

B. Learning Algorithm

Initial neuron parameters of SOMs are randomly set at the beginning of the learning process. Then the joint angle outputs will lead the end-effector to wrong positions and make the manipulator take inadequate poses. By giving an arbitrary target in the workspace, they update the parameters. After many times iteration of the updating, an appropriate relation between the image spaces and the joint angle space will be established.

For N-th iteration, the SOMs updates their parameters using the learning flow mentioned below.

1) A target u_t is arbitrarily given in the workspace. The target positions in the camera images are extracted and transferred to SOMs.

2) Each SOM sorts its neurons in the ascending order of the distances between the target and W of the neurons.

$$\left\|\mathbf{u}_t - \mathbf{W}^1\right\| < \cdots < \left\|\mathbf{u}_t - \mathbf{W}^{\lambda^n}\right\| \tag{6}$$

Here λ^n is the number of neurons that will be updated in the n-th iteration.
3) One SOM is chosen and outputs the joint angles $\boldsymbol{\theta}_0^{out}$ by (7). The manipulator moves using $\boldsymbol{\theta}_0^{out}$.

$$\boldsymbol{\theta}_0^{out} = \frac{\sum_{i=1}^{\lambda^n} g\left(order_i^n, \lambda^n\right)\left(\boldsymbol{\theta}_i + \mathbf{J}_i^{\dagger}\left(\mathbf{u}_t - \mathbf{W}_i\right)\right)}{\sum_{i=1}^{\lambda^n} g\left(order_i^n, \lambda^n\right)} \tag{7}$$

Here $order_i^n$ is the order of i-th neuron. Since appropriate neuron parameters have not been obtained yet, however, the joint angles lead the end-effector to a wrong position. Then the cameras obtain the new end-effector position v_0 .
4) The SOM improves the output by (8) so that it can reduce the positioning error.

$$\boldsymbol{\theta}_1^{out} = \boldsymbol{\theta}_0^{out} + \frac{\sum_{i=1}^{\lambda^n} g\left(order_i^n, \lambda^n\right)\left(\boldsymbol{\theta}_i + \mathbf{J}_i^{\dagger}\left(\mathbf{u}_t - \mathbf{v}_0\right)\right)}{\sum_{i=1}^{\lambda^n} g\left(order_i^n, \lambda^n\right)} \tag{8}$$

The cameras obtain the new end-effector position v_1.
5) The SOMs update their neuron parameters as following subsection.
6) The system iterates the above process for defined times.
C. Updating the Parameters
Each neuron parameters are updated using u_t, θ_{out}, and v.
1) Updating W
W is updated by (9).

$$\mathbf{W}_i^{n+1} = \mathbf{W}_i^n + \varepsilon_W^n \, g\left(order_i^n, \lambda^n\right)\left(\mathbf{u}_t - \mathbf{W}_i^n\right) \tag{9}$$

Here ε_W^n is the learning coefficient for W. It has a large value for early stages of learning and has a small value for late stages. By updating W the neurons will be distributed all over the image spaces.
2) Updating J
J is updated by (10).

$$\mathbf{J}_i^{n+1} = \mathbf{J}_i^n + \varepsilon_J^n \, g\left(order_i^n, \lambda^n\right)\Delta\mathbf{J}_i^n \tag{10}$$

Here ε_J^n is the learning coefficient for J, and it changes just like ε_W . $\Delta\mathbf{J}_i^n$ is determined by Widrow-Hoff's learning rule. The error function E_J is

$$E_J = \frac{1}{2}\left\|\left(\mathbf{v}_1 - \mathbf{v}_0\right) - \mathbf{J}_i^n\left(\boldsymbol{\theta}_1^{out} - \boldsymbol{\theta}_0^{out}\right)\right\|^2 \tag{11}$$

Then $\Delta\mathbf{J}_i^n$ is determined by

$$\Delta\mathbf{J}_i^n = -C_J \frac{\partial E_J}{\partial \mathbf{J}_i^n} = \frac{\left(\mathbf{v}_{01} - \mathbf{J}_i^n\,\boldsymbol{\theta}_{01}^{out}\right)\boldsymbol{\theta}_{01}^{out\,T}}{\left\|\boldsymbol{\theta}_{01}^{out}\right\|^2} \tag{12}$$

Here $\mathbf{v}_{01} = \mathbf{v}_1 - \mathbf{v}_0$, $\boldsymbol{\theta}_{01}^{out} = \boldsymbol{\theta}_1^{out} - \boldsymbol{\theta}_0^{out}$, and $C_J = 1/\left\|\boldsymbol{\theta}_{01}^{out}\right\|^2$. By updating J SOMs become to output more appropriate θ_{out}.
3) Updating ξ
ξ is the gradient vector of the evaluation function.

$$\boldsymbol{\xi} = \alpha_M \boldsymbol{\xi}_M + \alpha_O \boldsymbol{\xi}_O \tag{13}$$

Here ξ_M is the gradient vector of H_M, ξ_O is the gradient vector of H_O.
ξ_M is updated by (14).

$$\boldsymbol{\xi}_{M,i}^{n+1} = \boldsymbol{\xi}_{M,i}^n + \varepsilon_{\xi M}^n\, g\left(order_i^n, \lambda^n\right)\Delta\boldsymbol{\xi}_{M,i}^n \tag{14}$$

Here $\varepsilon_{\xi M}^n$ is the learning coefficient similar to ε_W^n . $\Delta\boldsymbol{\xi}_{M,i}^n$ is determined likely with $\Delta\mathbf{J}_i^n$. The error function is

$$E_{\xi M} = \frac{1}{2}\left\|\left(H_{M,k} - H_{M,j}\right) - \boldsymbol{\xi}_{M,i}^{n\;T}\left(\boldsymbol{\theta}_k^{out} - \boldsymbol{\theta}_j^{out}\right)\right\|^2 \tag{15}$$

Here j is the ID number of the neuron that is the closest to the target, and k is the ID number of the 2nd neuron. Then $\Delta\boldsymbol{\xi}_{M,i}^n$ becomes as following.

$$\Delta\boldsymbol{\xi}_{M,i}^n = -C_{\xi M}\frac{\partial E_{\xi M}}{\partial \boldsymbol{\xi}_{M,i}^n} = \frac{\left(H_{M,jk} - \boldsymbol{\xi}_{M,i}^{n\;T}\,\boldsymbol{\theta}_{jk}^n\right)\boldsymbol{\theta}_{jk}^{n\;T}}{\left\|\boldsymbol{\theta}_{jk}^n\right\|^2} \tag{16}$$

Here $H_{M,jk} = H_{M,k} - H_{M,j}$, $\boldsymbol{\theta}_{jk}^{n} = \boldsymbol{\theta}_{k}^{n} - \boldsymbol{\theta}_{j}^{n}$, and $C_{\xi M} = 1 / \left\| \boldsymbol{\theta}_{jk}^{out} \right\|^{2}$.
ξ_O is updated by (17).

$$\boldsymbol{\xi}_{O,i}^{n+1} = \boldsymbol{\xi}_{O,i}^{n} + \varepsilon_{\xi O}^{n} \, g\left(order_i^n, \lambda^n\right) \Delta \boldsymbol{\xi}_{O,i}^{n} \tag{17}$$

Here $\varepsilon_{\xi O}^{n}$ is the learning coefficient similar to ε_{W}^{n}. $\Delta \boldsymbol{\xi}_{O,i}^{n}$ is also determined likely with $\Delta \mathbf{J}_{i}^{n}$. The error function is

$$E_{\xi O} = \frac{1}{2} \left\| \left(H_{O,1} - H_{O,0} \right) - \boldsymbol{\xi}_{O,i}^{n\ T} \left(\boldsymbol{\theta}_{1}^{out} - \boldsymbol{\theta}_{0}^{out} \right) \right\|^{2} \tag{18}$$

Here $H_{O,0}$ and $H_{O,1}$ are respectively the potential values for v_0 and v_1. Then $\Delta \boldsymbol{\xi}_{O,i}^{n}$ becomes as following.

$$\Delta \boldsymbol{\xi}_{O,i}^{n} = -C_{\xi O} \frac{\partial E_{\xi O}}{\partial \boldsymbol{\xi}_{O,i}^{n}} = \frac{\left(H_{O,01} - \boldsymbol{\xi}_{O,i}^{n\ T} \boldsymbol{\theta}_{01}^{n} \right) \boldsymbol{\theta}_{01}^{n\ T}}{\left\| \boldsymbol{\theta}_{01}^{n} \right\|^{2}} \tag{19}$$

Here $H_{O,01} = H_{O,1} - H_{O,0}$, $\boldsymbol{\theta}_{01}^{n} = \boldsymbol{\theta}_{1}^{n} - \boldsymbol{\theta}_{0}^{n}$, and $C_{\xi O} = 1 / \left\| \boldsymbol{\theta}_{01}^{out} \right\|^{2}$.
By updating ξ using (14) and (17), the SOMs become to output joint angles that make the manipulator achieve high manipulability and take obstacle free poses.
4) Updating θ
θ is updated by (20).

$$\boldsymbol{\theta}_{i}^{n+1} = \boldsymbol{\theta}_{i}^{n} + \varepsilon_{\theta}^{n} \, g\left(order_i^n, \lambda^n\right) \Delta \boldsymbol{\theta}_{i}^{n} \tag{20}$$

Here ε_{θ}^{n} is the learning coefficient similar to ε_{W}^{n}. $\Delta \boldsymbol{\theta}_{i}^{n}$ is determined as

$$\Delta \boldsymbol{\theta}_{i}^{n} = \boldsymbol{\theta}_{i}^{Desire} - \boldsymbol{\theta}_{i}^{n} \tag{21}$$

$$\boldsymbol{\theta}_{i}^{Desire} - \boldsymbol{\theta}_{0}^{out} = \mathbf{J}_{i}^{n\dagger} \left(\mathbf{W}_{i}^{n} - \mathbf{v}_{0} \right) + \left(\mathbf{I} - \mathbf{J}_{i}^{n\dagger} \mathbf{J}_{i}^{n} \right) \boldsymbol{\xi}_{i}^{n} K_{p}^{n} \tag{22}$$

Here K_{p}^{n} is a positive coefficient, it is introduced to realize high manipulability and obstacle free poses. It also decreases with learning times in order to enable fine tuning of the system. Then $\Delta \boldsymbol{\theta}_{i}^{n}$ becomes as following.

$$\Delta \boldsymbol{\theta}_{i}^{n} = \boldsymbol{\theta}_{i}^{out} - \boldsymbol{\theta}_{i}^{n} + \mathbf{J}_{i}^{n\dagger} \left(\mathbf{W}_{i}^{n} - \mathbf{v}_{0} \right) + \left(\mathbf{I} - \mathbf{J}_{i}^{n\dagger} \mathbf{J}_{i}^{n} \right) \boldsymbol{\xi}_{i}^{n} K_{p}^{n} \tag{23}$$

By updating θ using (20), the SOMs become to output θ^{out} so that the manipulator moves its hand-effector with less error, that it achieves high manipulability, and that it takes obstacle free poses.

4. Path Planning

Collision avoidance in our system is realized on the basis of the following idea. While the path projections in any one camera images do not interfere with the obstacle projections, the path does not collide with the obstacle in 3-dimensional space. Also the SOMs determine joint angles of the manipulator so that it takes obstacle free poses for given target position. By combining the SOMs and a simple path planning system, then, we can realize collision avoidance. The path planning system only has to make a trajectory that does not collide with obstacles in the camera images. To control the manipulator by the SOMs outputs for points on the trajectory is to realize obstacle avoidance. The idea is different from most of existing algorithms that uses configuration space. Our planning system plans paths in 2-dimensional spaces and then the computational cost is lower than them. The system adopts Laplace potential method. The method can avoid local minima. For detail, please refer to our previous study (Han et al., 2006).

5. Simulation Results

We have constructed an experimental system. The outline of the system is shown in Fig.4. By simulation and experiments, we have also revealed that a visuo-motor system with 3 CCD cameras and 2 SOMs can control a redundant manipulator and realize collision avoidance in an environment with obstacles.

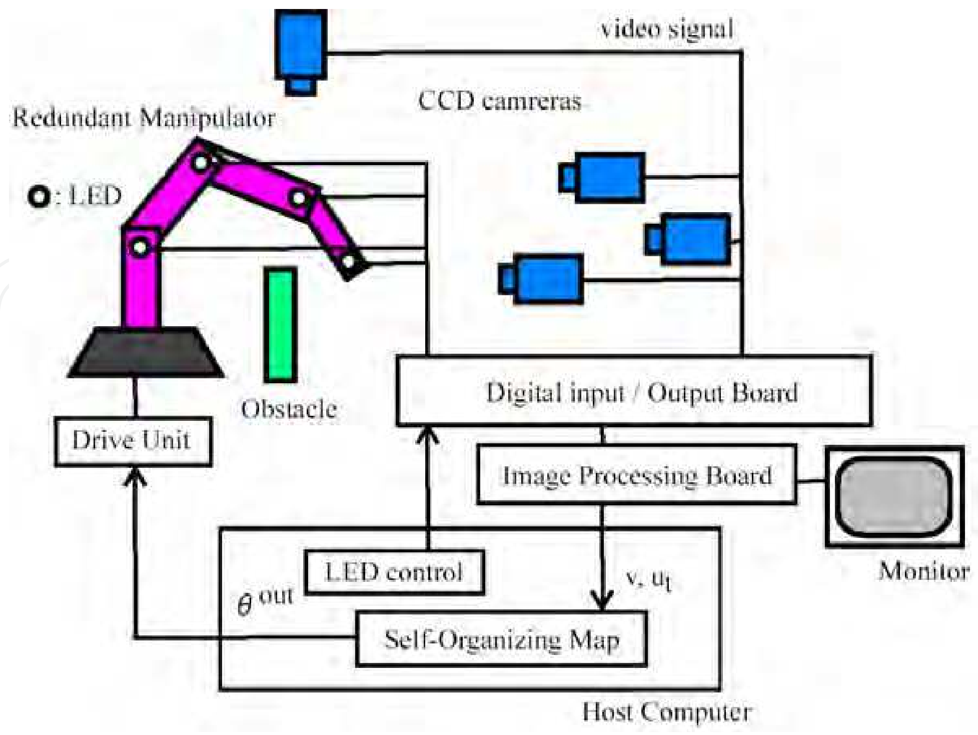

Fig. 4. Outline of our experimental system

In this paper, we aim at a system with 4 CCD cameras and a 4-DOF redundant manipulator, and show its validity by simulation. The cameras are assumed to be orthographic models and each camera has 640*480 pixel resolution. The simulation model is illustrated in Fig. 5. The length of each link is 120, 135, 110 and 140 pixels in the image spaces. Each SOM involves 240 neurons. 15000 targets were given in the learning process and they were distributed within 200*200 pixel rage with a focus on the obstacle. The time required for the learning was about 10 minutes using a PC with 3.0GHz Pentium4. The learning parameters are listed in Table 1.

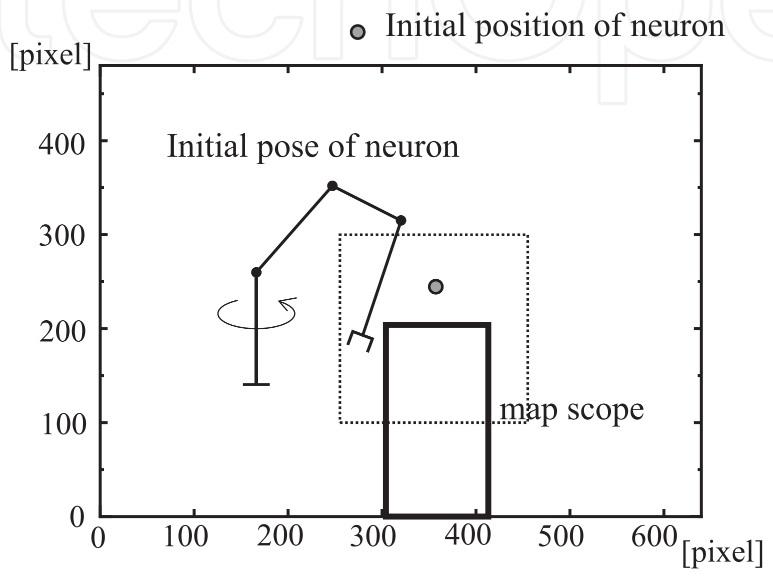

Fig. 5. Assumed simulation environment illustrated in the image space of the right camera

After the learning, 45 target positions were given randomly to evaluate the positioning error of the end-effector and to confirm the poses of the manipulator. The following figures show the positions of the end-effector and poses of the manipulator obtained by each camera. "x" marks are positions of the end-effector, and "o" are the projected targets.

	λ	ξ_W	ξ_O	K_p	α_1	α_2
Initial value	48	1.0	0.3	0.24	5.0	0.3
Final value	1	1.0	1.0	0.0012		

Table 1. Learning parameters used in the simulation

Fig.6 shows targets that are visible from all cameras and poses of the manipulator. It can be seen that there are some poses that touch the obstacle in the image spaces of side cameras. However, the manipulator does not collide with the obstacle with the poses indeed, since they are obstacle-free poses in the image space of the top camera. The figures also show that each SOM keeps continuousness with each other. The average positioning error of the end-effector was 1.89 pixels.

When both the initial and the goal positions can be observed in one side camera, the collision avoidance can be performed by using the only the corresponding SOM. An example of path planning is shown in Fig.7. The path planning system planned a collision-free path of the end-effector in the top camera image using Laplace potential method, and determined the shortest path in another camera image. Then the planning system divided the path into 34 positions and the SOMs outputted the collision-free poses for the positions. The figures show that the planned path avoided the obstacle. Also, they show that the system controlled the manipulator switching outputs from the SOMs in an environment where occlusion occurred.

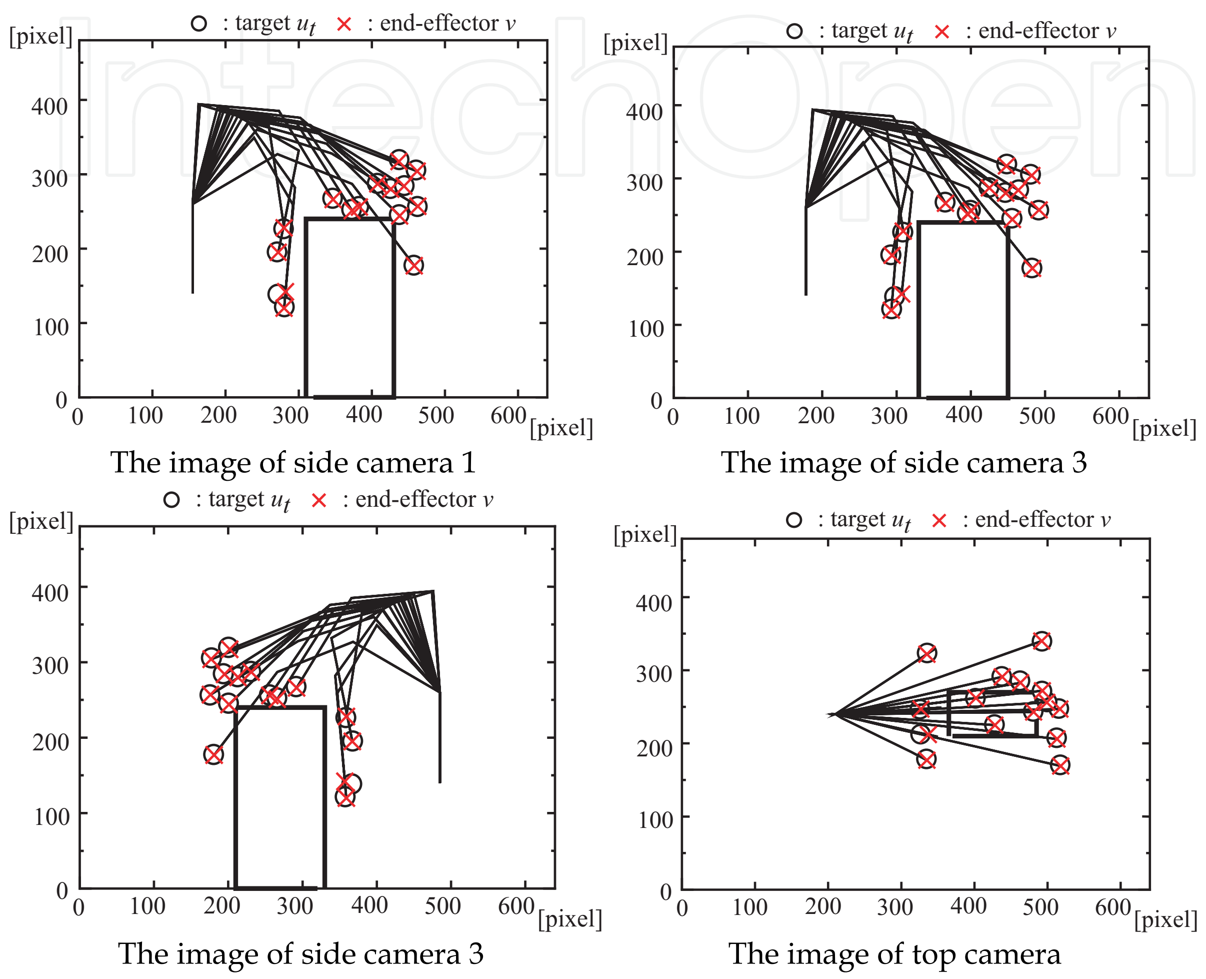

Fig. 6. Targets and manipulator poses after the learning process

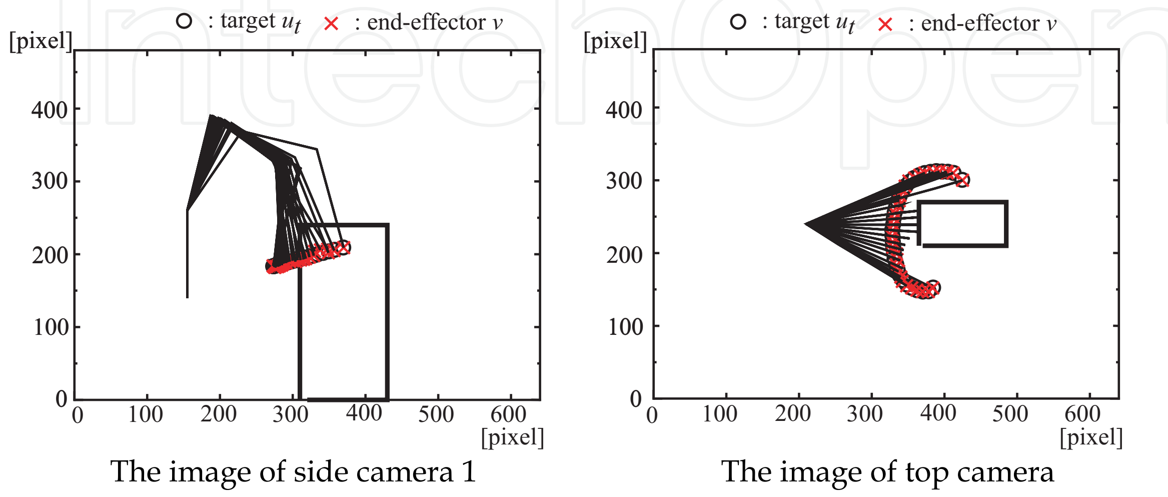

Fig. 7. Collision avoidance by the simulation

6. Conclusions

We developed a visuo-motor system with multiple self-organizing maps. The system consists of a redundant manipulator, multiple cameras and multiple SOMs corresponding to the cameras. By using the cameras, the system can control the manipulator in an environment with obstacles. To realize the system, we also developed a learning method of the SOMs. The learning method can keep consistency of outputs among the SOMs. We then combined the SOMs and a path planning system to achieve collision avoidance. Since the SOMs outputs collision free poses, the path planning system became very simple. Simulation results showed that the SOMs control the manipulator with obstacle free poses and that the collision avoidance was realized.

7. References

Behera, L. & Kirubanandan, N. (1999). A hybrid neural control scheme for visuo-motor coordination, *IEEE Control Systems*, pp.34-41.

Buessler, J. L.; Kara, R. ; Wira, P. ; Hihl, H. & Urban, J. P. (1999). Multiple self-organizing maps to facilitate the learning of visuo-motor correlations, *Proc. of the IEEE Int. Conf. on Systems, Man and Cybernetics*, pp.470-475.

Buessler, J. L. & Urban, J. P. (1998). Visual guided movements: learning with modular neural maps in robotics, *Neural Networks*, 11, pp.1395-1415.

Carusone, J. & Eleurterio, G. (1998). The feature CMAC: a neural-network based vision system for robotic control, *Proc. of the IEEE Int. Conf. on Robotics and Automation*, vol.4, pp.2959-2964.

Collobert, M. (2006). Using multi-kohonen self-organizing maps for modeling visual perception, *Computer Vision and Graphics*, pp.1031-1036.

Han, M.; Okada, N. & Kondo, E. (2003). Collision avoidance for a visuo-motor system using a self-organizing map in a 3D space, *6th Japan-France Congress on Mechatronics*, pp.495-500.

Han, M. ; Okada, N. & Kondo, E. (2006). Coordination of an uncalibrated 3-D visuo-motor system based on multiple self-organizing maps, *JSME International Journal*, Series C, vol.49, pp.230-239.

Kohonen, T. (1998). Self-organizing maps and associative memory, *Springer information Sciences*, vol.8, pp.43-48.

Marinetz, T. ; Ritter, H. & Schulten, K. (1990). Three-dimensional neural net for learning visuo-motor coordination of a robot arm, *IEEE Trans. on neural networks*, vol.1, pp.131-136.

Miller, W. (1989). Real-time application of neural networks for sensor-based control of robots with vision, *IEEE Trans. on Systems, Man and Cybernetics*, vol.19, No.4, pp.825-831.

Okada, N. ; Shimizu, Y. ; Maruki, Y. & Yoshida, A. (1999). A self-organizing visuo-motor map for a redundant manipulator in environment with obstacles, *Proc. of the 9th Int. Conf. on Advanced Robotics*, pp.517-522.

Walter, J. A. & Schulten, K. J. (1993). Implementation of self-organizing neural networks for visuo-motor control of an industrial robot, *IEEE Trans. on Neural Networks*, vol.4, pp.86-95.

Wiener, J.; Burwick, T. & von Seelen, W. (2000). Self-organizing maps for visual feature representation based on natural binocular stimuli, *Biological Cybernetics*, 82, pp.97-110.

Zeller, M. ; Sharma, R. & Schulten, K. (1997). Motion planning of a pneumatic robot using a neural network, *IEEE Control Systems*, 17, pp.89-98.

Zha,H. B.; Onitsuka, T. & Nagata,T. (1996). A visuo-motor coordination algorithm for controlling arm's movement in environments with obstacles, *Proc. of the 4th Int. Conf. on Control, Automation, Robotics and Vision*, pp.1013-1017.

Self-Organizing Maps
Edited by George K Matsopoulos

ISBN 978-953-307-074-2
Hard cover, 430 pages
Publisher InTech
Published online 01, April, 2010
Published in print edition April, 2010