Visual Reinforcement Learning Algorithm using Self Organizing Maps and Its Simulation in OpenGL Environment

Hiroshi Dozono Ryouhei Fujiwara and Takeshi Takahashi
Faculty of Science and Engineering
Saga University
1-Honjyo Saga 840-8502
JAPAN
hiro@dna.ec.saga-u.ac.jp fujiwara@dna.ec.saga-u.ac.jp takahasi@dna.ec.saga-u.ac.jp

Abstract: - Recently, the camera systems becomes more available for mobile robots . But scene analysis for generating control signals is still difficult and consumes large computational power. For this problem, the control method which generates the control signals directly from the raw camera images will be effective. In this paper, we use the reinforcement learning using the camera image as input data. For the division of the states represented with camera images, self organizing map is introduced. The division of the states and learning of the control signal using reinforcement learning are executed simultaneously on the map. For examining the performance of this algorithm, we made the simulation system with Graphical User Interface using OpenGL.

Key-Words: - Reinforcement Learning, Self Organizing Map, Learning algorithm, Mobile robot, OpenGL

1 Introduction

Research for the mobile robot produces communication robots used in families and cleaning robots used in offices. For the autonomous movement, the robot must get the information of environment, analyze it, select an action and execute the action. In this process, the first problem is how to recognize the environment.
As the conventional sensors for mobile robot, ultra sonic sensors, optical sensors and cameras are often used. From these devices, we select the camera device because visual information is rich. Additionally the camera devices become very popular for computer systems recently. Some notebook computers and mobile computers are integrated with the cameras and USB cameras become more outperformed and cheaper than before. In this situation, we can arrange the hardware of mobile robots which are controlled by visual information obtained from cameras and the software which captures the camera images can be easily developed regardless for the Operating System(OS)(Windows or Unix).

For controlling mobile robots, further software which handles the vision system intelligently is required. For example, the scene analysis should be applied to the image to detect the target, obstacles and other environmental objects. But, the intelligent scene analysis requires much computation powers for real-time processing.

For this problem, the applied mathematical methods which handle the image data directly are often used. In this paper, we propose a control method for mobile robots which uses the camera image directly as the input data of Reinforcement Learning(RL)[1]. RL is a machine leaning method which can obtain the control rules adapting to the environment according to the reward given for the actions of mobile robot. The division of the input space of environment is required for the RL, but for camera images the prior division of the input space is not given. For the grid spaces, the input space are simply divided to each grid, but for the camera images, the input space can not divided so simply because of the non-linear transformation from the coordinate system to camera images. For this problem, we introduce Self Organizing Map(SOM)[2] which automatically makes the division of input space simultaneously with RL of control system. As the control system using Self Organizing maps, Self Organizing Relationship network(SOR)[3] and Modular Network SOM(MN-SOM)[4] were reported. Both of them use SOM to organize the input vectors on the competitive layer of SOM. SOR and MN-SOM use fuzzy inference and neural network for generating control signals respectively. Compared with these methods, our algorithm is very simple, thus it will be possible to apply to the low-end computers or embedded systems.

For the experiments of mobile robots using neural networks or reinforcement learning, many iterations of learning with operating the robots are required and for each iteration, the robot should starts from random point in the operation space. Using the real mobile robots, these operations will take very long time. In this paper, we propose a method for the experiments of mobile robots using the operation space simulated using OpenGL[5]. The operation space is modeled in the computer and rendered by OpenGL. The camera image of the mobile robots is taken from the virtual camera and processed by the learning systems. Furthermore, we have developed the experiment system equipped with graphical user interfaces, with which the parameters of the SOM and RL can be modified easily and the camera image, the map and traces of the mobile robot are displayed in real time during the learning. Using this experiment system, we can examine the performance of the learning algorithm using visual information efficiently. In this paper, we made some experiments of Reinforcement Learning using Self Organizing Map on this system with changing the parameters and operation spaces of mobile robots.

2 Self Organizing Map(SOM) and Reinforcement Learning(RL)

SOM is an architecture of neural networks, which is classified as the network of feed forward type and of the unsupervised learning method. SOM can organize the feature of the input vectors on the 2-dimensional map on which the output neurons are arranged. After learning, the input vectors are mapped on the organized map, then the relations of the input vectors can be visualized on the map regardless to the dimension of input vector, thus SOM can be used for the analyses of very high dimensional vectors. Original SOM algorithm trains the map incrementally by updating the map for each presentation of input vector. The recent trend of SOM algorithm adopts Principal Component Analysis(PCA) and batch update to improve the performance, but in this paper, we use the conventional incremental learning SOM for incremental improvements of robot behaviors.

RL is a type of machine learning method based on the policy to select the action which leads more rewards. RL is a kind of supervised learning for which the correct answer to the input data is not given but the reward to the result concerning the action and the history of the actions is presented. A set of environmental states is given and the probability for taking each action at each state is learned according to the reward concerning the action. RL is often applied to the control of mobile robot. In conventional applications, the states are prior divided depending on the definitions of the environments,. For example, for the mobile robot using ultra sonic sensors, the environments is represented in the binary values translated from sensors inputs and for the robot working in the grid world, the coordinates of the grid represents the environments. For these cases, the divisions of the state are simple based on the immediate values of sensor inputs or coordinates.

3 Visual Reinforcement Learning using Self Organizing Map (SOM-RL)

As mentioned before, we propose a control method using raw image data based on RL for mobile robots . But the division of the states is not prior given for image data. For this problem, we use the SOM to divide the input space of visual images to the states represented by the images associated to the units on the competitive layer of SOM. For this purpose, we developed the algorithm of Reinforcement Learning using Self Organizing Maps(SOM-RL). We made some experiment of the clustering the images obtained from virtual camera simulated in the OpenGL world and confirmed that the images are clustered well using SOM.. The architecture of the network is shown in Fig.1

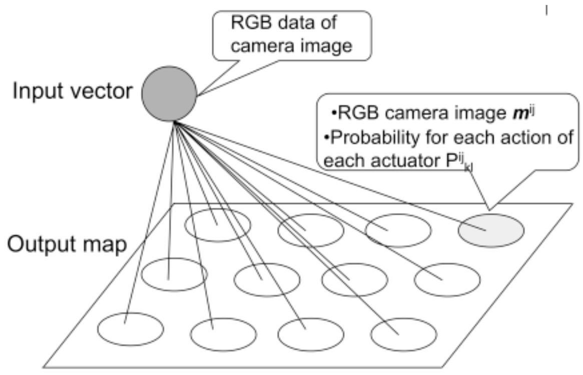

RGB data of camera image
Input vector
•RGB camera image m^{ij}
•Probability for each action of each actuator P^{ij}_{kl}
Output map

Fig.1 Architecture of the network of SOM-RL

The algorithm is shown as follows.

SOM-RL algorithm

Step 1 Initialization

Initialize all of $\mathbf{m}^{ij}$, q^{ij}_{kl} using random values, where $\mathbf{m}^{ij}$ is the vector(image) associated to the unit U^{ij} on the competitive layer and p^{ij}_{kl} is the probability of selecting l-th action for k-th actuator associated

to U^{ij}. Set the initial neighbor radius $R=R_{init}$, learning rate of SOM Ls=iLs learning rates of reinforcement learning Lr_1= iLr_1, Lr_2= iLr_2.
Set mobile robots at random place in the moving area.

Step 2 Moving the mobile robot
(1)Get the camera image and store in the vector **x**.
(2)Search for the best match unit $U^B=U^{ij}$ which is associated to the closest $\mathbf{m}^{ij}$ from **x**.
(3) Select the control signal c_k(1<= c_k =L) for each actuator A_j based on the probability p^{ij}_{kl} and move the robot. After moving robot, get the reward Rv from the environment

Step 3 Learning of the vector and probability
For U^B and each units $U^{i'j'}$ which are in the neighbor radius r from U^B,
(1) Update the vector $\mathbf{m}^{i`j`}$ associated to $U^{i`j`}$ as follows.

$$\mathbf{m}^{i`j`}= \mathbf{m}^{i`j`}+Ls*f(r)*(\mathbf{x}- \mathbf{m}^{i`j`}) \quad (1)$$

where f(r) is neighborhood function

and $r=\sqrt{(i-i`)^2+(j-j`)^2}$. (2)

(2) Update the probabilities $p^{i`j`}_{kl-}$ associated to $U^{i`j`}$ as follows.
For each k,

$$p^{i`j`}_{kl} = p^{i`j`}_{kl} + Rv \times Lr_1 \times f(r) \times \sum_{\substack{h=1\\ h\neq c_{,,k}}}^{L} p^{i`j`}_{kh}$$

for l= c_k

$$p^{i`j`}_{kl} = p^{i`j`}_{kl} - Rv \times f(r) \times Lr_1 \times p^{i`j`}_{kl}$$

for $l \neq c_k$ (3)

(3) If the number of move reaches MAXMOVE, goto Step 4
(4) If mobile robot successfully reaches goal, update the probabilities p^{ij}_{kl} as follows
For each U^{ij} used for successful move

$$p^{ij}_{kl} = p^{ij}_{kl} + Rv \times Lr_1 \times f(r) \times \sum_{\substack{h=1\\ h\neq c_{,,k}}}^{L} p^{ij}_{kh} \text{ for } l=c_k$$

$$p^{ij}_{kl} = p^{ij}_{kl} - Rv \times Lr_1 \times f(r) \times p^{ij}_{kl} \text{ for } l \neq c_k \quad (4)$$

and goto Step 4.
(5) Goto Step 2.

Step 4 Restart mobile robot
(1) Update Ls=Ls – dLs, Lr_1=Lr_1-dLr_1, Lr_2=Lr_2-dLr_2 , R=R-dR.

(2) Reset the mobile robot at random point in the moving area and goto Step 2 during the predefined iterations.

In Step 1, the parameters and the position of the robot is initialized. The initial values for the vectors and probabilities associated to the units are set randomly. In Step 2, the action of the mobile robot is selected using the camera image. The action is defined as the pattern of actuator control signals. The best match unit which is associated to the closest image to the camera image is searched and the action is selected depending on the probabilities associated to the unit. In Step 3, the vectors and probabilities are updated. Using conventional SOM update method, the vectors associated to the best match units and its neighboring units are updated. The probabilities associated to the best match unit and its neighboring units are also updated using the reinforcement learning at each step. And if the mobile robot reaches the goal, the successful path is updated again in (4). In Step 4, the parameters of the learning are updated and the robot is reset to random point during predefine iterations.

3 Experimental Result in Simple Projection Space.

Before applying SOM-RL using raw camera images, we tested SOM-RL algorithm in simple projection space. The working space of the mobile robot is set as 10.0x10.0 square area($0 \le x \le 10, -5 \le y \le 5$) and a ball is set at (1,0) in the space. For each movement of the mobile robot, the size and position of the ball are calculated using simple projection transformation and are used for the input vector of SOM. The robot has 4 actuators, which enables the robot to move any direction directory. The velocities of 4 actuators are set proportional to Cv(t) which is the decreasing value with learning iterations and the directions are set based on the model of real robot as Table 1.

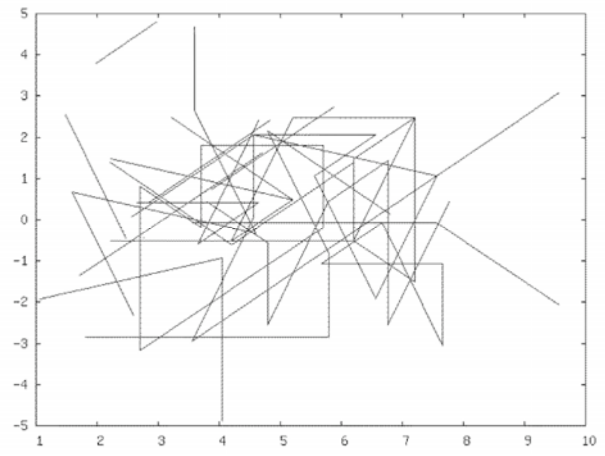
Table 1 Velocities of the actuators for each control pattern

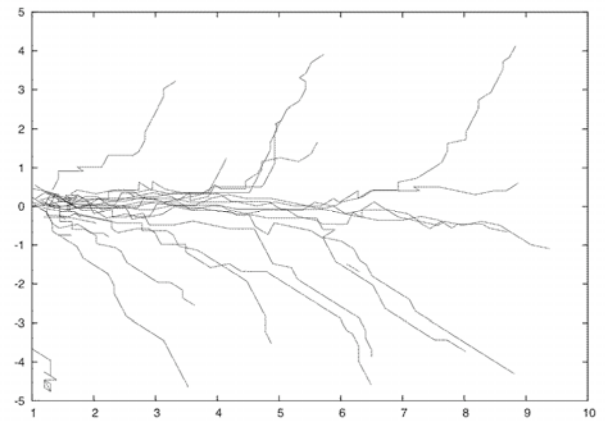
	Control Signals		
	Pattern 1	Pattern 2	Pattern 3
actuator 1	Vx=1, Vy=1	Vx=0, Vy=0	Vx=-1, Vy=-1
actuator 2	Vx=1, Vy=-1	Vx=0, Vy=0	Vx=-1, Vy=1
actuator 3	Vx=-1, Vy=-1	Vx=0, Vy=0	Vx=1, Vy=-1
actuator 4	Vx=-1, Vy=-1	Vx=0, Vy=0,	Vx=1, Vy=-1

For each actuator, one of the pattern of control signal is selected at each movement.
To simplify the situation, the angle of the camera is fixed to horizontal direction to the goal object, so the camera can always capture the goal object. The reward is given by the following equation.

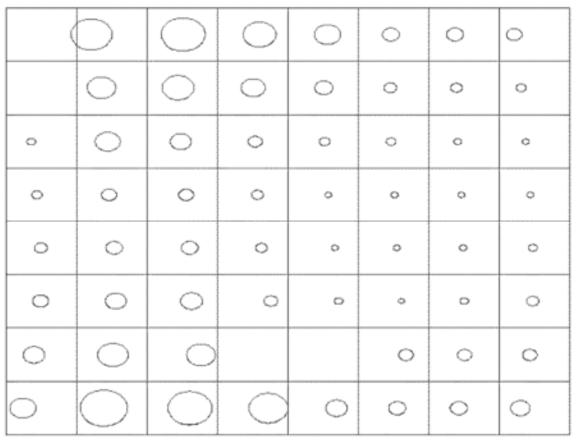
$$Rv = \begin{cases} 1 & (abs(y_c(t-1)) - abs(y_c(t)))G - (1-G)(s_c(t) - s_c(t-1)) > 0 \\ -1 & otherwise \end{cases} \tag{5}$$

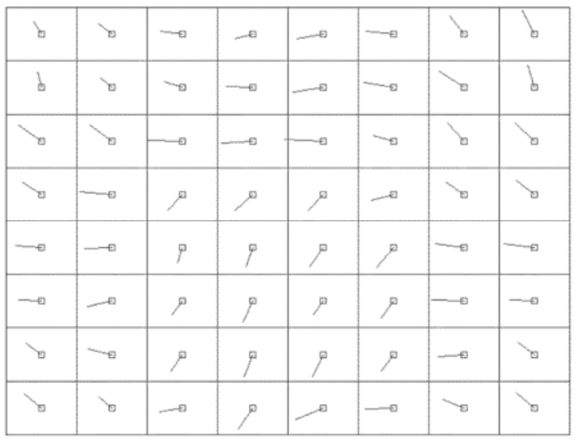
where $y_c(t)$ and $s_c(t)$ are y coordinate and size of goal object in the camera image respectively. 4 actuators are set to the diagonal direction of x-y coordinate system. The mobile robot is reset to random point if the robot move to left side of the line x=1. The parameters of SOM-RL are set as iLs=0.8, iLr_1=0.5, iLr_2=0 and the map size is 8x8. Number of the iteration is set as 10000 move of the robots. Fig.2 shows the traces of the mobile robots in beginning and ending of learning. The position of the goal is (0,0) on the graph.


The traces of the robots in beginning of learning


The traces of the robots in ending of learning
Fig.2 The traces of mobile robots in projection space(1)

In the beginning of learning, the robots move almost randomly because the initial values of the probability set as random. In the ending of learning, the robots move toward the position of goal objects. Fig.3 shows the map of the image of goal object and average of the weighted sum of actuator output for each position of the map.
The similar images of the goal objects are mapped closely on the map and the actuator output learned by RL also shows that similar output patterns are mapped closely on the map.


Map of the camera image of goal object


Map of the average of weighted sum of actuator output
Fig.3 Map organized by SOM-RL(1)

Next, we changed velocities of actuators nonuniformly as Table 2.

Table 2 Velocities of actuators for nonuniform setting

	Control Signals		
	Pattern 1	Pattern 2	Pattern 3
actuator 1	Vx=0.5, Vy=0.25	Vx=0.5, Vy=−0.25	Vx=−1, Vy=−1
actuator 2	Vx=0.75, Vy=−0.75	Vx=−0.25, Vy=0.25	Vx=−1.25, Vy=0.75
actuator 3	Vx=−0.5 Vy=−1.25	Vx=0.5, Vy=0.3	Vx=0,75 Vy=1.5
actuator 4	Vx=−0.75, Vy=−1.5	Vx=0,25 Vy=−0,25	Vx=1.25, Vy=−0.5

In this case, the selection of the control patterns is difficult even for human. Fig.4 shows the traces of the robot after learning.

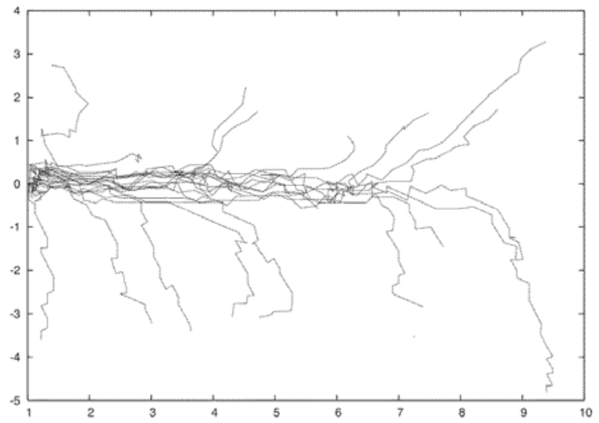


Fig.4 Traces of the robot for nonuniform actuator settings

The robot also moves toward the goal successsfully. From these experiments, SOM-RL can learn the control signal of mobile robot with classifying the input vectors to the units which correspond to the states of RL.

In the previous experiments, the direction of the camera is fixed. But, it is not realistic assumption. The camera should be fixed to the robot and the direction of the camera should change with the rotation of the robot. In this case, the robot can not identify its position using the size and position of the goal object in the image. For this problem, we set one more reference object which is located near the goal object and uses the size of the goal object, position of the goal object and position of the reference object in the camera image as input vectors for SOM. The robot can identify its position by triangulation. Fig.5 shows the traces of the roboot.

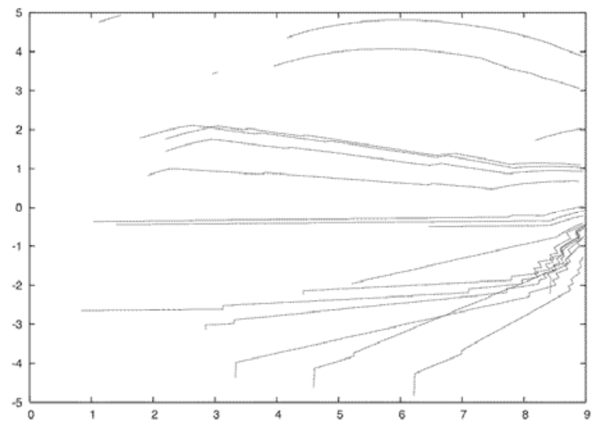


Fig.5 Traces of the robot using reference object.

In this case, the position of the goal is located (9,0). For some cases, the robot moves toward the goal, but for other cases, the robot moves toward right and can not reach the goal. The information of goal object and reference object is not enough for learning the control signal using camera images. For thie problem, the use of full image information will be improve the situation. In the next section, the experiment using full images under OpenGL environments will be discussed.

4 Simulation of the Mobile Robot using OpenGL

OpenGL is the cross platform API for 2D and 3D graphics available for Windows, Unix, Linux, MacOS and even for embedded systems like mobile phones. The source codes of C programs are compatible for these systems. Our system is tested on WindowsXP, Ubuntu Linux and MacOSX Leopard. It can draw the simple 2D and 3D object like cuboid, ball and cylinder in any colors using any lightings. Camera image can be taken from any positions with setting arbitrary camera parameters. To simulate more realistic spaces, texture mappings of the real space can be applied to the objects in the model.

In this paper, we used the operation space which consists of the simple objects labeled by colors. Each wall is labeled with identical color and the obstacles and goal are also labeled with color. The drawing size of OpenGL is set to 256x256. A sample of the camera image is shown in Fig.6.

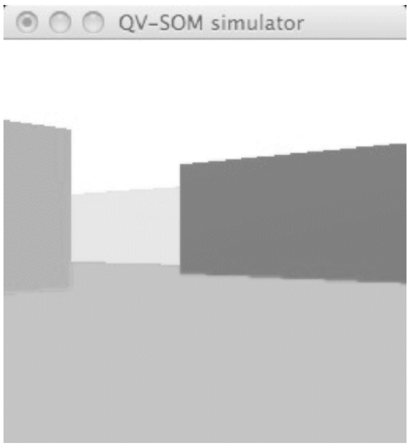
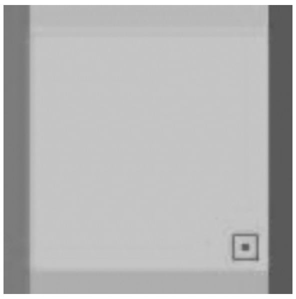


Fig.6 Sample of camera image drawn by OpenGL

After drawing each frame, the frame buffer of the OpenGL Window is copied to the 3 dimensional array(color,X,Y) and culled to 32x32 to decrease the dimension of the vector given to SOM. The culled image is directly used as the input vector of SOM, so size of the input vector is 32x32x3=3072.
We made experiment system equipped with Graphical User Interface (GUI) for Windows XP Operating System. Fig.7 shows the main window of this system.
On the left of the window, some dialogs used for modifying the parameters of the SOM and RL, and the file name for description of operation space are arranged. The operation space can be given by the simple text file which describe the position, size and color of goal, obstacles and walls. On the right side of the window, the camera image, traces of the robot and map of the learned images are displayed in real time during the learning. Using this system, the process of the learning is confirmed visually during the learning.

5 Experimental Result in OpenGL Space

Next, we made the experiments using OpenGL. At first, we made experiments with changing the number of the iterations of learning to examine the sufficient number of the iterations. We used the following operation space shown in Fig.8.



Space-1

Fig.8 Simple operation Space

The point surrounded by square at right bottom is goal object colored in red. For all spaces, the

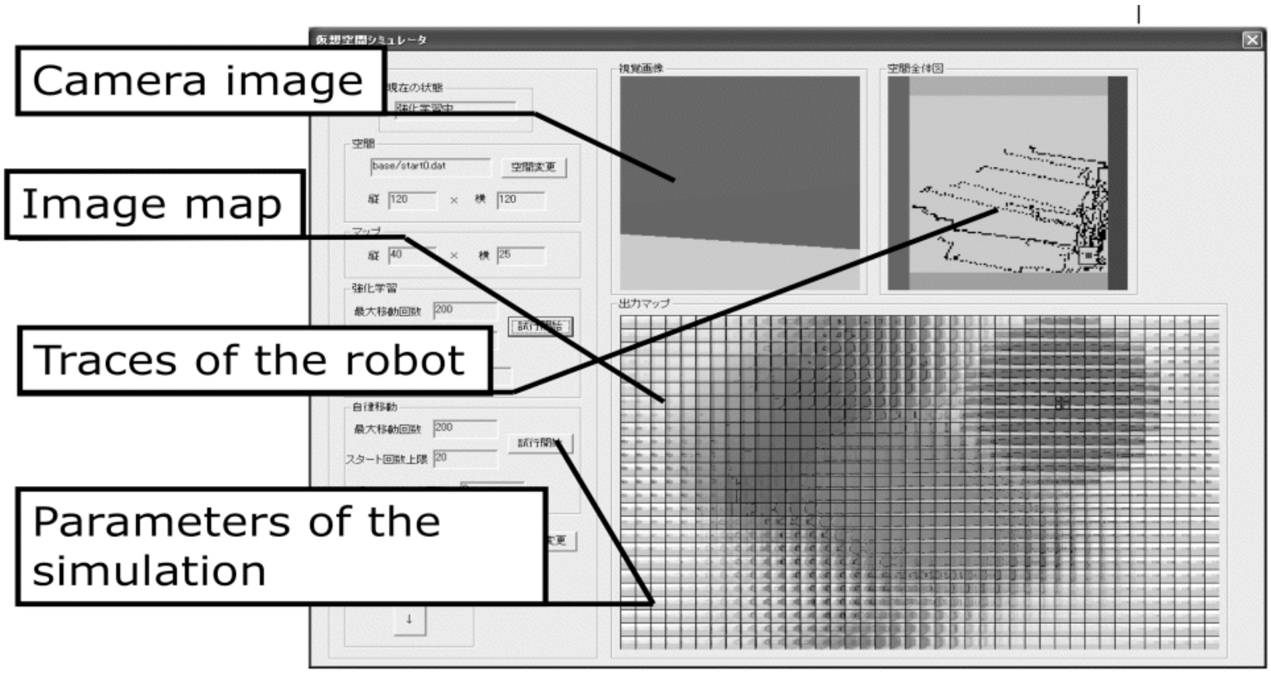
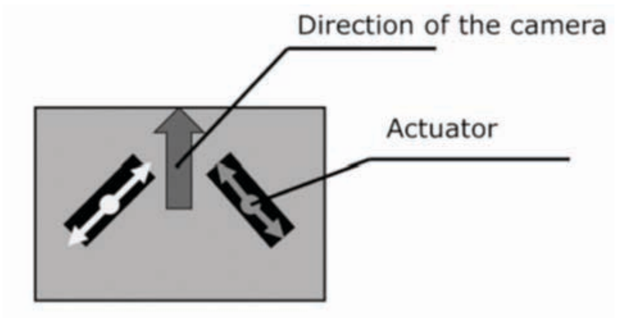


Fig.7 Main window of Simulation system

surrounding 4 walls are colored in the different colors. There are no obstacles in space-1. The size of the space is 100x100. 2 actuators are equipped to the left and right side of the mobile robot and the actions of each actuator are forward, backward and stop and the speed is 1 for each step. The model of the robot is shown in Fig.9



Fig,9 Model of the mobile robot

The velocity of the each actuator is shown in Table 3.

Table 3 Velocities of actuators for OpeGL space simulation

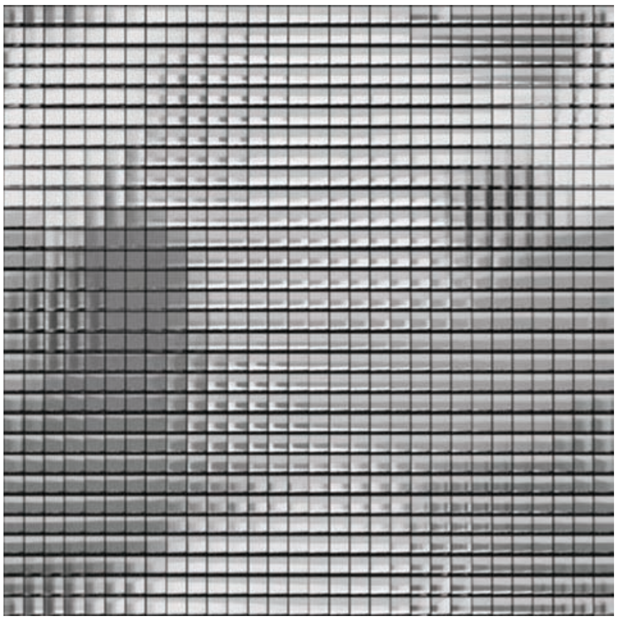
	Control Signals		
	Pattern 1	Pattern 2	Pattern 3
actuator 1	Vx=1, Vy=1	Vx=0, Vy=0	Vx=-1, Vy=-1
actuator 2	Vx=1, Vy=-1	Vx=-0, Vy=0	Vx=-1, Vy=1

The rewards are defined as follows.

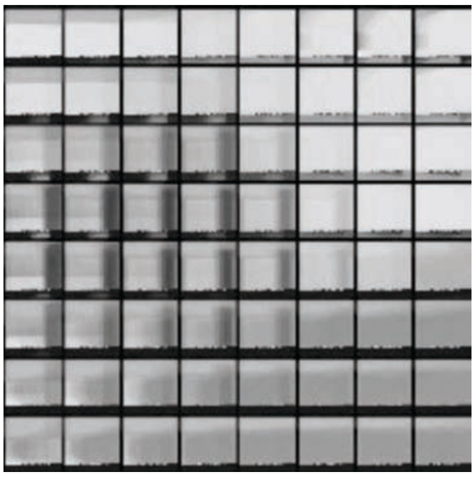
For Step 3(2),
If the robot moves closer to the goal, Rv=Rv+1
For Step 3(4)
If the robot reach to the goal, Rv=1

The parameters of the learning are iLs=0.8, iLr_1=0.5, iLr_2=0.7, MAXMOVE=300 and size of the map is 30x30. The number of iterations are set as 100, 300, and 500. For each iteration of trial. the robot moves until whether the robot reach the goal or moves MAXMOVE times. Fig.10 shows all of the map and part of 8x8 units on the map organized by the learning of space-1.

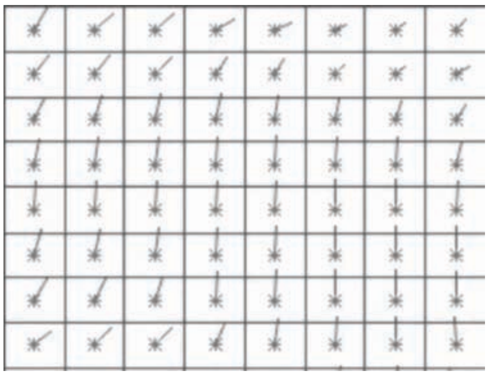
The similar images are organized closely on the map and the actuator outputs are continuously changing on the map. After learning the maps, 100 or 50 iterations of the tests of moving mobile robot are made. Table 4 shows the number of successful goals for each number of iterations of learning.



Map of the camera images



Magnified map of the images



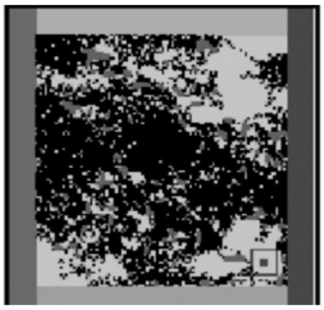
Magnified map of average weighted sum of actuator output

Fig.10 Map organized by SOM-RL(2)

Table 4 Number of successful goals for each number of learning iterations

Number of iterations	0	100	300	500
Number of goals	3/100	89/100	92/100	94/100

The successful number of goals increases according to the number of iterations, but 300 iterations is considered to be sufficient for learning of this space, so the number of iteration is fixed to 300 for the following experiments. Fig.11 shows the traces of the robots for before learning and after learning of 300 iterations.


before learning

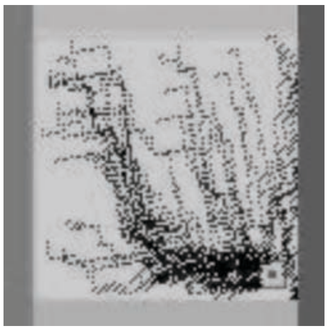
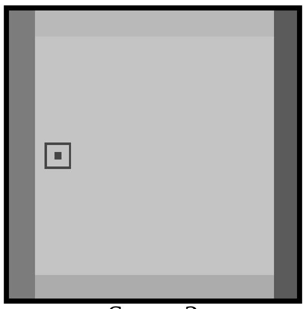
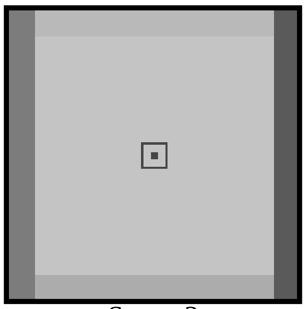

after 300 iterations

Fig. 11 Traces of the robots in OpenGL spaces

Before learning, the robot moves around the starting points, but after learning, robot moves toward the goal point.
Next, we examined the effects of the positions of the goals. Fig.12 and Table 5 shows the operation spaces with changing the position of goals and the number of successful goals respectively.


Space 2


Space 3

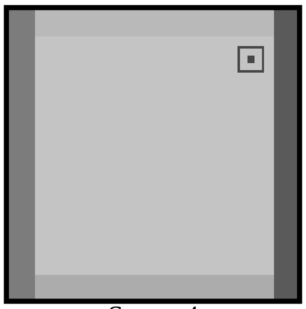

Space 4

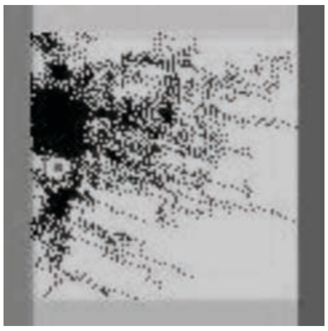
Fig.12 Operation spaces with changing the goal positions

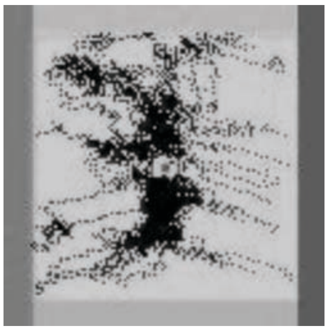
Table 5 Number of successful goals with changing the goal position

For each space, the number of successfull goals

	Space 2	Space 3	Space 4
Before learning	3/50	6/50	0/50
After learning	50/50	50/50	47/50

increases apparently after learning. But, the succesfull rates are different depending on the goal positions. It is considered that the average distance from random position to the goal shorter in space 2 and space3 become shorter compared with the fixed number of movements. Fig.13 shows the traces of robot for each space.


Space 2


Space 3

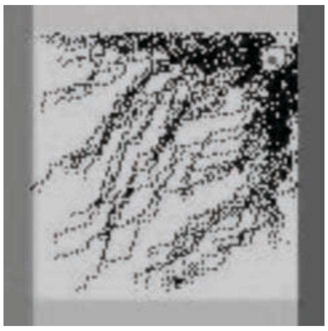
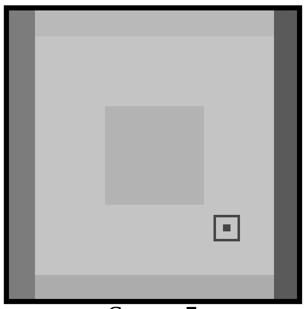
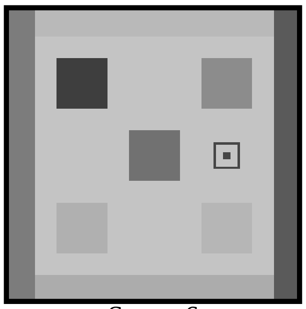

Space 4

Fig.13 Traces of the robot with changing the position of the goal

For all space, the robot moves toward the goal positions. But, the traces are gather at some positions in space 2 and space 3. It is the effect of neighborhood functions used to update the images and probabilities in SOM-RL algorithm.
Next, the effects of the obstacles are examined with setting the different arrangement of the obstacles. Fig.14 shows the operation spaces.


Space 5


Space 6

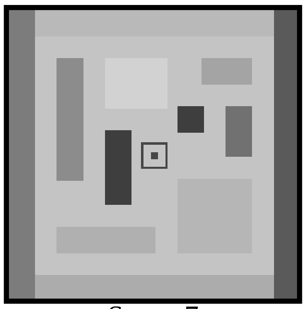
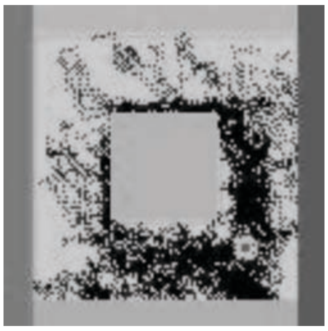
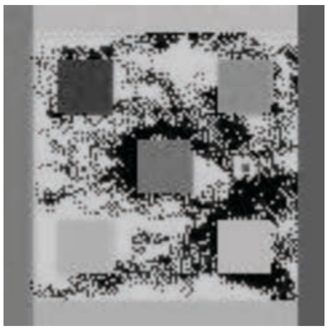

Space 7

Fig.14 Operation spaces with different arrangements of the obstacles

Fig.15 and Table 6 shows the traces of the robot and number of the successfull goals for each space respectively. For each arrangements of the obstacles, the number of successfull goals increases after learniing. But, the rates of the increments are different depending on the arrangements of the obstacles. The more complex arrangements of the obstacles lead the less number of successfull goals as expected. .


Space 5


Space 6

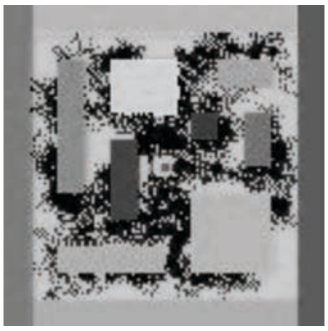

Space 7

Fig.15 Traces of the robots with different arrangements of the obstacles

Table 6 Number of the successfull goals with different arrangement of the obstacles

	Space 5	Space 6	Space 7
Before learning	1/50	2/50	3/50
After learning	41/100	36/50	36/50

Next, to examine the effects of colors of the wall and obetacles, some experiments are made with setting the colors of the walls and obstacles in mono-color. Table 7 shows the result
The number of successfull goals is apparently decrease compared with the previous experiments. For this method, the different color of the walls and obstacles have significant role to make the robot to identify the environments. In the real world, the walls and obstacles have textures(wall paper, shelves, windows), so it will be possible to distinguish the position of walls

Table 7 Number of successfull goals for mono-color walls and obstacles

	Space 1'	Space 3'	Space 5'
Before learning	3/50	1/50	1/50
After learning	18/50	3/50	9/50

6 Conclusion

In this paper, we propose a reinforcement learning algorithm using self organizing maps for state division. This algorithm can be used for the reinforcement learning using the camera images for input data. The performance of this algorithm is examined in simple projection space and in OpenGL space. This algorithm can generate the control signal for the movements of the mobile robot that can avoid simple obstacles, but more improvement will be needed for avoiding complex arrangement of obstacles.

As the future works, this algorithm must be tested in more realistic spaces. Using the texture mapping of the photograph to walls and obstacles in OpenGL spaces, the operating space becomes much more realistic. Furthermore, this algorithm will be applied to the real robot after learning the control signals in texture mapped OpenGL spaces.

References:

[1] R.S.Sutton and A.G.Barto, Reinforcement learning An intrduction, MIT Press, 1998
[2]] T. Kohonen: Self Organizing Maps, Springer, 2004
[3] T.Koga, K.Horio and T.Yamakawa, The self-organizing relationship (SOR) network employing fuzzy inference based heuristic evaluation, Neural networks: 2006 Special issue: Advances in self-organizing maps--WSOM'05, Vol.19, No.6, 2006, pp.799-811
[4] Tetsuo Furukawa, Modular Network SOM and Self-Organizing Homotopy Network as a Foundation for Brain-like Intelligence MM-SOM,
Proceedings of the 6th international Workshop on Self Organizing maps, 2007
[5] D. S. Mason Woo, and J. Neider, OpenGL(R) Programming Guide: The Official Guide to Learning OpenGL(R), Addison-Wesley